Non-Gradient Hash Factor Learning for High-Dimensional and Incomplete Data Representation Learning

Di Wu, *Member, IEEE,*, Shihui Li, Yi He, *Member, IEEE,*, Xin Luo *Fellow, IEEE*, and Xinbo Gao *Fellow, IEEE*

***Abstract*—High-dimensional and incomplete (HDI) data are ubiquitous in various big data-related industrial applications, such as drug innovation and recommender systems. Hash-learning is the most efficient representation learning approach to extract hidden information from HDI data owing to its fast reasoning and low storage. However, an existing hash learning approach commonly employs gradient-based optimization techniques to address the discrete objective caused by the binary nature of hash factors, where the *Quantization* (i.e., quantizing the real values to binary codes) loss is inevitable, resulting in accuracy loss when representing HDI data. Motivated by these critical and vital issues, this paper proposes a non-gradient hash factor (NGHF) model with three-fold ideas: a) innovating a discrete differential evolution (DDE) algorithm able to simulate the continuous optimization via disabling bits of binary codes based on the projected Hamming dissimilarity, thus enabling an effective discrete optimizer, b) applying the proposed DDE algorithm to directly optimize the discrete learning objective of NGHF defined on HDI data, thereby facilitating its efficient and precise training without any *Quantization* loss, and c) theoretically proving the convergence of NGHF. As such, NGHF possesses high representation learning ability comparable to that of a real-valued model, making it able to achieve precise binary representation to HDI data. Extensive experimental results on nine real-world datasets demonstrate that NGHF significantly outperforms eight state-of-the-art hash learning models. Moreover, its accuracy is amazingly comparable to that of a real-valued model for HDI data representation learning. Such results are inspiring for facilitating hash-learning models with both high accuracy and fast reasoning on HDI data, which is critical for industrial applications. Our source code is shared at the link: https://github.com/wudi1989/NGHF.**

***Index Terms*—Hash Learning, Representation Learning, Missing Data Estimation, High-dimensional and Incomplete data, Differential Evolutionary, Latent Factor Analysis.**

I. Introduction

MATRICES are commonly adopted to describe the relationships between two types of entities in various big data-related industrial applications such as recommender systems and drug networks [1–4]. For example, the matrix located at the top left corner of Fig. 1 describes the relationships between users and items for recommender systems, where each row indicates a specific user, each column indicates a specific item, and each entry records a user's preference for an item. Notably, such a matrix from industrial applications is usually high-dimensional and incomplete (HDI) because it is difficult to fully observe all the relationships between a large number of entities [5, 6]. Thus, the crux of extracting hidden information from HDI matrices lies in how to achieve effective and efficient representation learning to them.

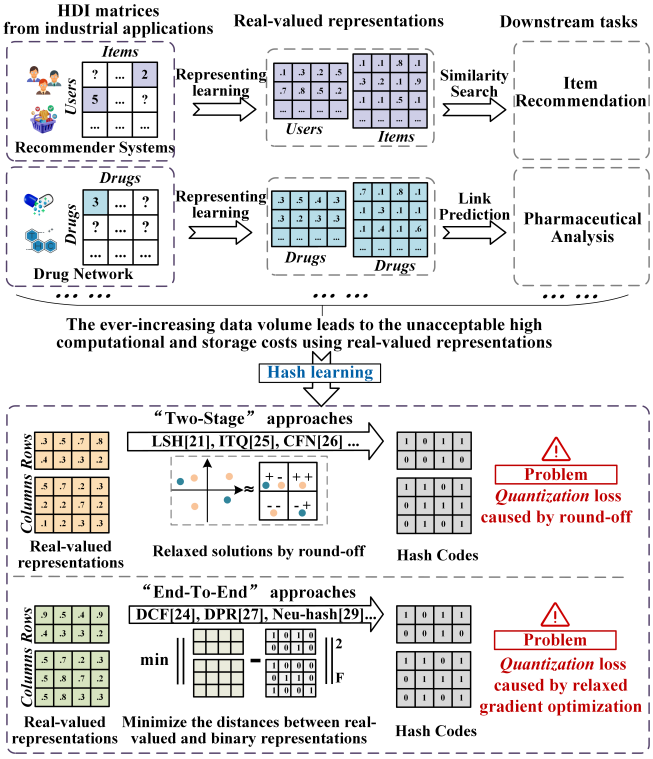


Fig. 1: The illustrations of representing large-scale HDI matrices. The upper panel shows that the real-valued model has unacceptably high computational and storage costs. The lower panel shows that existing hash learning approaches have the problem of *Quantization* loss.

Latent factor (LF) model [7–10], originating from matrix factorization methods, is one of the most effective representation learning approaches designed for HDI matrices. It maps an HDI matrix into the low-rank latent space to learn the real-valued representations of entities [13]. These representations can be utilized to implement downstream tasks, such as item recommendation based on similarity search and pharmaceuti-

✦ Di Wu, Shihui Li, and Xin Luo are with the College of Computer and Infor-mation Science, Southwest University, Chongqing 400715, China (e-mail: wudi.cigit@gmail.com, lishihui1008@email.swu.edu.cn, luoxin21@gmail.com).
✦ Yi He is with the Department of Data Science, William & Mary, Williamsburg 23186, VA, USA (e-mail: yihe@wm.edu).
✦ Xinbo Gao is with the State Key Laboratory of Integrated Services Networks, School of Electronic Engineering, Xidian University, Xi'an 710071, Shaanxi, China (e-mail: xbgao@mail.xidian.edu.cn).
✦ Corresponding Authors: Xin Luo and Xinbo Gao.

cal analysis based on link prediction as shown in the upper panel of Fig. 1. Nevertheless, such real-valued representations have extremely high computational and storage costs when the number of entities is very large [17], e.g., a recommender system usually has millions of users and items). Hence, the LF model's efficiency is unacceptable in representing large HDI matrices from big data-related applications.

Fortunately, hash learning [17–20] can be applied to solve the LF model's inefficient problem by learning the binary representations in Hamming Space. Compared with real-valued representations, reasoning over binary representations using bitwise operations (e.g., Hamming distance) not only significantly reduces memory consumption but also achieves much faster computation [17, 18]. Currently, existing hash learning approaches can be categorized into two categories, i.e., "two-stage" and "end-to-end" approaches, as shown in the lower panel of Fig. 1. Two-stage approaches separate the optimization and quantization stages, where the discrete constraints are discarded in optimization and quantization is done by simple round-off [17], resulting in large quantization loss. To avoid simple round-offs, end-to-end approaches directly learn binary representations by minimizing the distances between real-valued and binary representations with relaxed gradient optimization [26]. However, end-to-end approaches still suffer from quantization loss, as they ultimately rely on converting real-valued representations into binary codes. Therefore, a serious challenge remains: can a hash learning model be built to learn binary representations without *Quantization* loss?

To avoid the *Quantization* loss that exists in gradient optimization-based hash learning approaches, this paper proposes a non-gradient hash factor (NGHF) model with three-fold ideas. First, a discrete differential evolution (DDE) algorithm is proposed to simulate the continuous optimization by disabling bits of binary codes based on the projected Hamming dissimilarity, thus enabling an effective discrete optimizer. Second, the proposed DDE algorithm is applied to directly optimize the discrete learning objective of the NGHF model defined on HDI data, thereby facilitating its efficient and precise training with little accuracy loss. Third, we theoretically prove the convergence of the proposed NGHF model. As such, NGHF possesses high representation learning ability comparable to that of a real-valued model, making it capable of achieving precise binary representation for HDI data. The main contributions of this work include:

a) This is the first study to design a non-gradient DDE algorithm to directly solve the discrete optimization of binary representation learning without *Quantization* loss.
b) A novel NGHF model is proposed, which can effectively and efficiently represent large-scale HDI matrices.
c) Theoretical analyses, convergence proofs, algorithm designs, and time complexity analyses are provided for the proposed NGHF model.

Extensive experiments on nine real HDI datasets are conducted to evaluate NGHF in scenes of hash learning. The results validate that NGHF significantly outperforms eight state-of-the-art hash learning models and has a comparable ability to a real-valued model in representing HDI data.

II. Related work

A. Latent Factor Model

The basic LF model was first proposed for implementing the recommender system [13]. Then, the LF model has been widely used in various industrial scenarios due to its advantages of high prediction accuracy and scalability [10–13]. So far, many models have been developed. For example, a federated FLF model was proposed for collaborative ranking with privacy-preserving [9]. A non-negative and symmetric LF model was proposed to estimate the data for a symmetric matrix [11]. The graph-regularized model [12] leverages graph-based techniques to capture the underlying structure of the data and improve prediction accuracy. Recently, several state-of-the-art LF models were proposed based on some advanced techniques. For example, the multi-metric LF models [10, 14] employ several norms to enhance a single-metric LF model. Lv et al. developed a semi-supervised technique to build a non-negative and symmetric LF model for community detection [15]. The multilayer-structured LF model was designed by a prediction sampling strategy for accurately representing HDI data [16].

Despite the emergence of various LF models, all of the previously mentioned ones are based on real values. With the explosive growth of data volume, their computational and storage costs become very significant in reasoning between a large number of entities. In comparison, the proposed NGHF model adopts the binary hash factors to represent an HDI matrix, which can significantly reduce the computational and storage costs of an LF model.

B. Hash Learning

Hash learning was initially applied to accelerate large-scale image and multimedia retrieval [44, 45]. Recently, it has received increasing attention from recommender systems [23–26]. In general, there are two kinds of hash learning, i.e., "two-stage" and "end-to-end" approaches. In terms of "two-stage" approaches. Das et al. [21] adopted locality-sensitive hashing [22] to generate binary codes for Google News users based on item-sharing similarity. Karatzoglou et al. [23] obtained binary codes by randomly projecting the learned user-item real-valued LFs. Zhou and Zha [17] generated binary codes by simply rounding off the gradient-optimization-learned real-valued hash factors. Zhang et al. [24] quantized real-valued LFs to obtain hash factors by ranking the order of the inner product rather than the similarities between users and items. Liu et al. [25] confidently applied uncorrelated bit constraints to the traditional collaborative filtering objective to effectively learn user-item LFs. However, these approaches learn hash representations through a continuous representation learning stage followed by a binary quantization stage [26], which introduces significant quantization errors.

To alleviate quantization errors, "end-to-end" approaches were developed. Zhang et al. [26] proposed discrete collaborative filtering (DCF) for optimizing hash representations directly. Later, discrete personalized ranking (DPR) [27] was proposed as a method for collaborative filtering with implicit feedback. Lightweight federated recommendation (LightFR)

was proposed for efficient privacy-preserving recommendation [48]. To aggregate the merits of both discrete and real-valued representations, a hybrid compositional coding for collaborative filtering (CCCF) model was proposed [28]. Han-sen et al. [30] proposed a new method called projection hamming dissimilarity, which measures dissimilarity between entities in the Hamming space with binary weighting of each dimension. Recently, by combining with various types of neural networks, deep hashing methods have been developed, such as Neuhash [29], HS-GCN [31], and the out-of-vocabulary method [49]. Moreover, graph neural networks were also adopted for capturing the complex relationships between users and items [46, 47, 50]. Guan et al. proposed a novel bi-directional heterogeneous graph hashing scheme for efficient personalized outfit recommendation [46, 47]. Chen et al. devised a bipartite graph contrastive hashing (BGCH+) method utilizing graph convolutional networks to improve Top-N search [50].

Nevertheless, these approaches employ gradient-based integer optimization, which still suffers from quantization loss due to the discretization process. Significantly different from them, the proposed NGHF model designs a non-gradient DDE algorithm to directly solve the discrete optimization of binary representation learning, which has no quantization loss.

III. PRELIMINARIES

A. LF Analysis on an HDI Matrix

Table I summarizes the primary symbols and notations adopted in this paper. We firstly recall the definitions below.

TABLE I: Symbols and Notations.

Symbol	Description
U, I	Two types of entity set.
u, i	The single element of U, I, $\forall u \in U$, $\forall i \in I$.
R	An HDI matrix to record the relationships between U and I.
Λ, Γ	The known and unknown elements in R, respectively.
D	The dimension of hash factor space.
$r_{u,i}$	An element of X at u-th row and i-th column.
$\hat{r}_{u,i}$	The corresponding approximation of $r_{u,i}$.
X, Y	Real-valued LF matrices.
x_u, y_i	u-th and i-th row-vector of X and Y respectively.
$\mathrm{z}_u, \mathrm{z}_i$	Two D-dimensional binary vectors.
W, Q	Hash factor matrices of a NGHF model.
w_u, q_i	u-th and i-th hash row-vector of W and Q respectively.
λ	The regularization controlling hyperparameters.
NP	The total number of individuals in the population.
g	The current generation of the DDE algorithm.
G	The maximum generation of the DDE algorithm.
$\mathbf{X}_g$	A binary discrete population at g-th generation.
$\mathbf{x}_k^g$	The k-th target vector (individual) at g-th generation, $k \in \{1, \ldots, NP\}$.
F_k	The mutation factor for k-th target vector.
m_k^g	The mutant-vector w.r.t. $\mathbf{x}_k^g$.
t_k^g	The new trial-vector w.r.t. $\mathbf{x}_k^g$.
t	Training iteration count.
$T_{\max}$	The maximum training iteration.

***Definition* 1 (*An HDI Matrix*)** Given two enclosed entity sets U and I and a matrix $R \in \mathbb{R}^{|U|\times|I|}$, where $|U|$ and $|I|$ are the cardinalities of U and I. R's each element $r_{u,i}$ denotes the quantification of a certain relationship between entity u ($u \in U$) and entity i ($i \in I$). Let Λ and Γ denote the enclosed sets of known and unknown elements in R, and $|\Lambda|$ and $|\Gamma|$ denote their cardinality, respectively. R is an HDI matrix when $|\Lambda| \ll |\Gamma|$.

Representing an HDI matrix denotes to mapping it into a low-dimensional latent space to learn informative latent representations [32, 33]. Consequently, an original HDI matrix can be easily analyzed based on the learned low-dimensional latent representations for downstream tasks, e.g., item recommendation and pharmaceutical analysis, as shown in the upper panel of Fig. 1. An LF model is one of the most popular methods for representing HDI matrices due to its high accuracy and ease scalability [14].

***Definition* 2 (*An LF Model*)** An LF model is to learn the latent representations of R based on Λ. Assuming the dimension of low-dimensional latent space is $D \ll \min\{|U|, |I|\}$. Let $x_u \in \mathbb{R}^D$ and $y_i \in \mathbb{R}^D$ be the representation vectors of an entity u ($u \in U$) and an entity i ($i \in I$), respectively, and $\hat{r}_{u,i}$ be the approximation of $r_{u,i}$. $\hat{r}_{u,i}$ is obtained by the inner product between x_u and y_i, i.e., $\hat{r}_{u,i} = x_u y_i^T$. Then, the representation vectors of x_u and y_i can be learned by minimizing the distances between $r_{u,i}$ and $\hat{r}_{u,i}$. After all the known elements of R (i.e., Λ) is adopted to learn the representation vectors of the entities of U and I, two LF matrices $X = [x_1, \ldots, x_u, \ldots, x_{|U|}] \in \mathbb{R}^{|U|\times D}$ and $Y = [y_1, \ldots, y_u, \ldots, y_{|I|}] \in \mathbb{R}^{|I|\times D}$ are obtained. Then, X and Y are the desired latent representations w.r.t. U and I, respectively. Note that X and Y are real-valued.

Definition 2 shows that an LF model requires a loss function to minimize the distances between the approximations and original elements in Λ. By adopting the squared loss to measure such distances [14, 33], an LF model's objective function is formulated as follows:

$$\arg\min_{X,Y} \sum_{r_{u,i}\in\Lambda} (r_{u,i} - \hat{r}_{u,i})^2 = \arg\min_{X,Y} \sum_{r_{u,i}\in\Lambda} \left(r_{u,i} - x_u y_i^{\mathrm{T}}\right)^2. \tag{1}$$

B. Projected Hamming Dissimilarity

Hash learning can significantly reduce data storage and computational costs of an LF model because it maps an HDI matrix into the Hamming space rather than the real-valued space to perform Boolean operations on the bit level. Hamming distance is widely adopted to calculate similarity in Hamming space as follows:

$$\begin{aligned} \delta_H(\mathrm{z}_u, \mathrm{z}_i) &= \sum_{d=1}^{D} \mathbf{1}[\mathrm{z}_{u,d} \neq \mathrm{z}_{i,d}] = \mathrm{SUM}(\mathrm{z}_u \,\mathrm{XOR}\, \mathrm{z}_i), \\ &\text{s.t. } \mathrm{z}_u \in \{+1, -1\}^D, \quad \mathrm{z}_i \in \{+1, -1\}^D. \end{aligned} \tag{2}$$

where z_u and z_i are two binary vectors, $\mathrm{z}_{u,d}$ denotes the d-th element in binary vector z_u, $\delta_H(\cdot)$ is the Hamming distance function, and $\mathrm{SUM}(\cdot)$ denotes the computed summation using the bit string instruction, respectively. Nevertheless, Eq.(2) still has a weakness in that each bit dimension is equally weighted, so that cannot measure the importance of each bit. To overcome such a weakness, a projected hamming dissimilarity is defined as follows.

***Definition* 3 (*Projected Hamming Dissimilarity* [30])** Given two entities u and i, let $\mathrm{z}_u \in \{+1, -1\}^D$ and $\mathrm{z}_i \in \{+1, -1\}^D$ be the binary hash factors of u and i, respectively, where D is

the dimension of Hamming space. $\delta_{PH}(\cdot)$ denotes the projected Hamming dissimilarity function. The projected Hamming dissimilarity between z_u and z_i is computed as follows:

$$\begin{aligned}\delta_{PH}(\mathrm{z}_u,\mathrm{z}_i) &= \|\mathrm{z}_u - P_{\mathrm{z}_u}(\mathrm{z}_i)\| \\ &= \mathrm{SUM}\Big(\mathrm{z}_u \,\mathrm{XOR}\, \underbrace{(\mathrm{z}_u \,\mathrm{AND}\, \mathrm{z}_i)}_{\text{projection}}\Big), \\ \text{s.t. } \mathrm{z}_u &\in \{+1,-1\}^D, \quad \mathrm{z}_i \in \{+1,-1\}^D.\end{aligned} \tag{3}$$

where $P_{\mathrm{z}_u}(\mathrm{z}_i) = \mathrm{z}_u$ AND z_i is a projection operator to project z_i onto z_u for allowing a binary importance weighting of z_i.

IV. The Proposed NGHF Model

The proposed NGHF model boasts two significant advantages over related studies [17, 26–31]. First and foremost, NGHF designs a DDE algorithm to directly solve the discrete optimization of binary representation learning without any Quantization loss. Second, the designed DDE algorithm weights each bit-dimension differently to identify a more critical bit-dimension, making NGHF represent an HDI matrix in a more informative manner.

Next, the proposed NGHF model is introduced in detail, which consists of five parts. The first part is 'objective formulation of NGHF', which introduces how to develop a hash-based LF model with projected hamming dissimilarity. The second part is 'discrete differential evolution (DDE) algorithm', which introduces how to improve the traditional differential evolution algorithm to achieve discrete optimization. The third part is 'optimizing the NGHF model with the DDE algorithm', which introduces how to directly optimize the objective function of the hash-based LF model without quantization loss. The fourth part is 'convergence proof', which discusses the convergence of NGHF. The final part is 'algorithm design and complexity analysis', which provides the pseudo-code of NGHF and analyzes its computational complexity.

A. Objective Formulation of NGHF

Similar to the LF model, the hash LF model still reconstructs each known element $r_{u,i}$ of an HDI matrix R but in the Hamming space rather than the real-valued space. Then, the approximation $\hat{r}_{u,i}$ of $r_{u,i}$ reconstructed by w_u and q_i with projected Hamming dissimilarity is defined as follows:

$$\begin{aligned}\hat{r}_{u,i} &= \frac{1}{D}\Big(D - \delta_{PH}(w_u, q_i)\Big) \\ &= 1 - \frac{1}{D}\mathrm{SUM}\Big(w_u \,\mathrm{AND}\, (\mathrm{NOT}\, q_i)\Big), \\ \text{s.t. } w_u &\in \{+1,-1\}^D, \quad q_i \in \{+1,-1\}^D.\end{aligned} \tag{4}$$

where $w_u\mathrm{AND}(\mathrm{NOT}q_i)$ is equivalent to $w_u\mathrm{XOR}(w_u\mathrm{AND}q_i)$ for speeding up the projected Hamming dissimilarity because $(\mathrm{NOT}q_i)$ can be pre-computed and stored to replace the original binary codes. To model a NGHF model, Eq.(1) is revised as follows:

$$\begin{aligned}&\arg\min_{W,Q} \sum_{r_{u,i}\in\Lambda} (r_{u,i} - \hat{r}_{u,i})^2 = \\ &\sum_{r_{u,i}\in\Lambda}\left[r_{u,i} - \Big(1 - \frac{1}{D}\mathrm{SUM}\big(w_u \,\mathrm{AND}\, (\mathrm{NOT}\, q_i)\big)\Big)\right]^2 \\ &\text{s.t. } W \in \{+1,-1\}^{|U|\times D}, \quad Q \in \{+1,-1\}^{|I|\times D}.\end{aligned} \tag{5}$$

where $W \in \{+1,-1\}^{|U|\times D}$ and $Q \in \{+1,-1\}^{|I|\times D}$ are the desired hash factor matrices, and w_u and q_i are the u-th and i-th row-vector of W and Q, respectively. In addition, following prior studies [26], balance constraints can maximize the information entropy of the bit. Thus, the balance constraints are required as follows:

$$\sum_{d=1}^{D} w_{u,d} = 0, \sum_{d=1}^{D} q_{i,d} = 0. \tag{6}$$

where $w_{u,d}$ and $q_{i,d}$ denote the d-th element of w_u and q_i, respectively. However, Eq.(6) is so strict and may make the model infeasible. Then, the balanced constraint is softened to be a regularization term. Accordingly, incorporating Eq.(6) into Eq.(5) as a regularization term as follows:

$$\begin{aligned}\arg\min_{W,Q} = &\sum_{r_{u,i}\in\Lambda}\left[r_{u,i} - \Big(1 - \frac{1}{D}\mathrm{SUM}(w_u \,\mathrm{AND}\, (\mathrm{NOT}\, q_i))\Big)\right]^2 \\ &+ \lambda\left(\left\|\sum_u w_u\right\|^2 + \left\|\sum_i q_i\right\|^2\right) \\ &\text{s.t. } W \in \{+1,-1\}^{|U|\times D}, \quad Q \in \{+1,-1\}^{|I|\times D}.\end{aligned} \tag{7}$$

where λ is the regularization controlling hyperparameter. Such a regularization term of Eq.(7) not only maximizes the information entropy of the bit but also avoids overfitting.

Following [13, 14, 16], an LF model is usually trained in a way of entry-by-entry manner on an HDI matrix. Then, the corresponding objective function of the NGHF model on a single element $r_{u,i} \in \Lambda$ is defined as follows:

$$\begin{aligned}\arg\min_{w_u,\, q_i} = &\left[r_{u,i} - \Big(1 - \frac{1}{D}\mathrm{SUM}\big(w_u \,\mathrm{AND}\, (\mathrm{NOT}\, q_i)\big)\Big)\right]^2 \\ &+ \lambda\left(\left(\sum_{d=1}^{D} w_{u,d}\right)^2 + \left(\sum_{d=1}^{D} q_{i,d}\right)^2\right) \\ &\text{s.t. } w_u \in \{+1,-1\}^D, \quad q_i \in \{+1,-1\}^D.\end{aligned} \tag{8}$$

Next, an efficient and accurate discrete optimization technique, i.e., the designed DDE algorithm is introduced to solve such an objective function of Eqs.(7)-(8).

B. The Discrete Differential Evolution (DDE) Algorithm

Let $\mathbf{X}^{NP\times D} \in \{+1,-1\}$ be a binary discrete population with *NP* individuals, G be the maximum evolution generation, g be the current generation, and $\mathbf{X}_g$ be the population of the current generation. In $\mathbf{X}_g$, each individual is a D-dimensional target-vector $\mathbf{x}_k^g = [\mathbf{x}_{k,1}^g, \ldots, \mathbf{x}_{k,d}^g, \ldots, \mathbf{x}_{k,D}^g]$, $\mathbf{x}_{k,d}^g \in \{+1,-1\}$ and k is an index between $[1, NP]$. Moreover, each individual has a score called fitness in a specific problem, which is calculated according to the fitness function [34-35]. The DDE algorithm consists of three phases, i.e., *mutation, crossover, and selection*.

*1) **Mutation Phase**:* An appropriate mutation strategy is crucial for the success of differential evolution in solving specific problems [36, 37]. Different mutation strategies can be used to generate mutant individuals for each individual $\mathbf{x}_k^g$, including *DE/Rand*/1, *DE/Rand*/2, *DE/RandToBest*/1, *DE/RandToBest*/2, *DE/Best*/2, and *DE/Best*/1 [34–37]. Next,

DE/Best/1 is chosen as a specific case to illustrate the DDE algorithm. *DE/Best*/1 generates the mutant-vector w.r.t. $\mathbf{x}_k^g$ as follows:

$$m_k^g = \mathbf{x}_{\text{best}}^g + F_k\left(\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g\right) \tag{9}$$

where m_k^g is the mutant-vector w.r.t. $\mathbf{x}_k^g$, $\mathbf{x}_{\text{best}}^g$ is the best individual of $\mathbf{X}^g$ in terms of the fitness function, r_1 and r_2 are mutually exclusive integers randomly generated within the range $[1, NP]$ and also different from the index k, F_k is the mutation factor. By following prior studies [37, 38], F_k can be set adaptively as follows:

$$F_k = \begin{cases} \text{SFGSS} & \text{if } rand_3 < \tau_2, \\ \text{SFHC} & \text{if } \tau_2 \le rand_3 < \tau_3, \\ F_l + F_u \cdot rand_1 & \text{if } rand_2 < \tau_1 \text{ and } rand_3 > \tau_3, \\ F_k & \text{if } rand_2 \ge \tau_1 \text{ and } rand_3 \ge \tau_3. \end{cases} \tag{10}$$

where F_k is initialized to a random value in the range $[0, 1]$, rand_1, rand_2 and rand_3 are pseudo-random numbers uniformly distributed between $[0, 1]$, τ_1, τ_2, τ_3, SFGSS, SFHC, F_l, and F_u are constants and are recommended to set as follows: SFGSS = 8, SFHC = 20, $F_l = 0.1$, $F_u = 0.9$, $\tau_1 = 0.1$, $\tau_2 = 0.03$, and $\tau_3 = 0.07$ [33–35].

It is worth noting that mutant-vector m_k^g is not a binary vector generated by Eq.(9). To generate m_k^g in the binary space, the following processing is performed:

$$m_{k,d}^g = \begin{cases} +1, & \text{if } m_{k,d}^g > \text{mean}(m_k^g), \\ -1, & \text{if } m_{k,d}^g \le \text{mean}(m_k^g), \end{cases} \tag{11}$$

where $\text{mean}(\cdot)$ is a function to calculate the average value of all elements in m_k^g and $m_{k,d}^g$ is the d-th element of m_k^g.

*2) **Crossover Phase**:* Since the values of individuals are either 1 or -1, the arithmetic crossover strategy is not suitable. Therefore, binomial crossover [37] is chosen to generate a new trial-vector t_k^g as follows:

$$t_{k,d}^g = \begin{cases} m_{k,d}^g, & \text{if } rand(0,1) \le CR \text{ or } d = d_{\text{rand}}, \\ \mathbf{x}_{k,d}^g, & Otherwise. \end{cases} \tag{12}$$

where CR is a preset constant within the range $[0, 1)$, d_{rand} is a random index that is generated within the range $[1, D]$ to ensure that trial-vector t_k^g gets at least one element from mutant-vector m_k^g, and $t_{k,d}^g$ is the d-th element of t_k^g.

3) Selection Phase: This phase is to decide which individual between trial-vector t_k^g and target-vector m_k^g could stay in the population of next-generation $\mathbf{X}_{g+1}$. The selection is decided by the fitness function [37, 38]:

$$\mathbf{x}_k^{g+1} = \begin{cases} t_k^g, & \text{if fitness}(t_k^g \mid \theta) \text{ is better than fitness}(\mathbf{x}_k^g \mid \theta), \\ \mathbf{x}_k^g, & \text{otherwise.} \end{cases} \tag{13}$$

where $\text{fitness}(\cdot \mid \theta)$ is the fitness function designed for the optimization objectives with parameter θ. Next, the DDE algorithm is applied to solve Eq. (8).

C. Optimizing the NGHF Model with the DDE Algorithm

To solve the discrete optimization problem of Eq. (7), the designed DDE algorithm is adopted with an alternating optimization strategy. Fig. 2 shows the processes of training

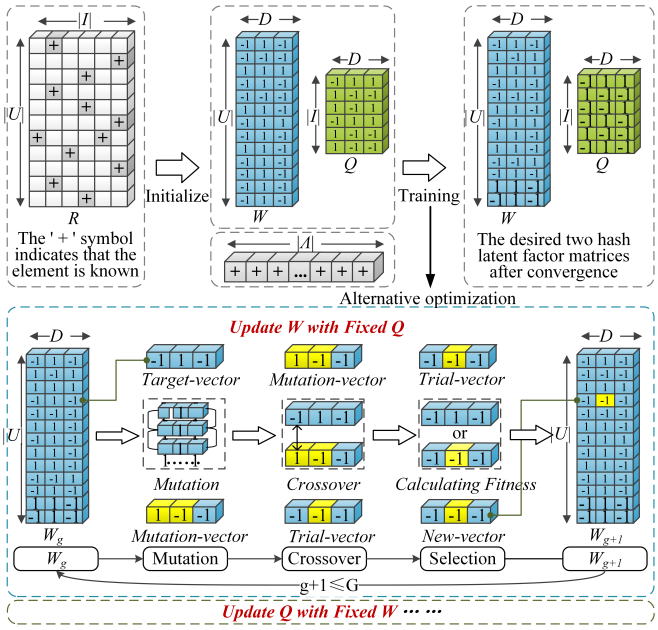


Fig. 2: The optimization process of the NGHF model with the DDE algorithm.

and optimization of the NGHF model. First is to randomly initialize two initial hash factor matrices $W \in \{+1, -1\}^{|U| \times D}$ and $Q \in \{+1, -1\}^{|I| \times D}$. Then, W and Q are iteratively updated until convergence. In each training iteration t, W and Q are updated alternatively by the DDE algorithm. Next, the updating processes of W and Q are elaborated at a training iteration t.

Let W be a population with $|U|$ individuals, in which each individual at generation g is composed of a binary hash factor vector $w_u^g = [w_{u,1}^g, \ldots, w_{u,d}^g, \ldots, w_{u,D}^g]$, $w_{u,d}^g \in \{+1, -1\}$, $d \in \{1, 2, \ldots, D\}$, and $u \in \{1, 2, \ldots, |U|\}$. Similarly, let Q be a population with $|I|$ individuals, in which each individual at generation g is composed of a binary vector $q_i^g = [q_{i,1}^g, \ldots, q_{i,d}^g, \ldots, q_{i,D}^g]$, $q_{i,d}^g \in \{+1, -1\}$, and $i \in \{1, 2, \ldots, |I|\}$. To optimize the objective function Eq. (7), the fitness function of the DDE algorithm on a single element $r_{u,i} \in \Lambda$ is designed based on Eq. (8) as follows:

$$\begin{aligned} &\text{fitness}\,(w_u \mid q_i) \text{ or fitness}\,(q_i \mid w_u) \\ &= \left[r_{u,i} - \left(1 - \tfrac{1}{D}\text{SUM}\,(w_u \text{ AND } (\text{NOT } q_i))\right)\right]^2 \\ &+\lambda\left(\left(\textstyle\sum_{d=1}^{D} w_{u,d}\right)^2 + \left(\textstyle\sum_{d=1}^{D} q_{i,d}\right)^2\right), \\ &\text{s.t. } w_u \in \{+1, -1\}^D,\ q_i \in \{+1, -1\}^D. \end{aligned} \tag{14}$$

where $\text{fitness}(w_u \mid q_i)$ or $\text{fitness}(q_i \mid w_u)$ is the fitness function of DDE algorithm w.r.t. w_u and q_i on a single element $r_{u,i} \in \Lambda$. With such a fitness function of Eq.(14), W and Q can be updated alternatively at a training iteration t as follows:

1) Update W with Fixed Q

a) Input W into the DDE algorithm as the initial population;
b) Initialize $g = 1$, $u = 1$;
c) Take one individual of W as the target-vector $w_u^g = [w_{u,1}^g, \ldots, w_{u,d}^g, \ldots, w_{u,D}^g]$, $w_{u,d}^g \in \{+1, -1\}$;

d) Generate mutation-vector $m_u^g = [m_{u,1}^g, \ldots, m_{u,d}^g, \ldots, m_{u,D}^g]$ according to Eqs. (9)-(11);
e) Generate trial-vector $t_u^g = [t_{u,1}^g, \ldots, t_{u,d}^g, \ldots, t_{u,D}^g]$ according to Eq. (12);
f) Separately calculate fitness($w_u^g \mid q_i$) and fitness($t_u^g \mid q_i$) according to Eq.(14);
g) Select the better one between w_u^g and t_u^g as the next generation of w_u^g according to Eq.(13);
h) Continue evolving the next target vector w_u^g, $u = u + 1$, until all individuals in the current generation g have evolved.
i) Repeat c)-h) to evolve the next generation population of W, $g = g + 1$.

2) Update Q with Fixed W

a) Input Q into the DDE algorithm as the initial population;
b) Initialize $g = 1$, $i = 1$;
c) Take one individual of Q as the target-vector $q_i^g = [q_{i,1}^g, \ldots, q_{i,d}^g, \ldots, q_{i,D}^g]$, $q_{i,d}^g \in \{+1, -1\}$;
d) Generate mutation-vector $m_i^g = [m_{i,1}^g, \ldots, m_{i,d}^g, \ldots, m_{i,D}^g]$ according to Eqs.(9)-(11);
e) Generate trial-vector $t_i^g = [t_{i,1}^g, \ldots, t_{i,d}^g, \ldots, t_{i,D}^g]$ according to Eq.(12);
f) Separately calculate fitness($q_i^g \mid w_u$) and fitness($t_i^g \mid w_u$) according to Eq.(14);
g) Select the better one between q_i^g and t_i^g as the next generation of q_i^g according to Eq.(13);
h) Continue evolving the next target vector q_i^g, $i = i+1$, until all individuals in the current generation g have evolved.
i) Repeat c)-h) to evolve the next generation population of Q, $g = g + 1$.

D. Convergence Proof

This section discusses the convergence of NGHF. As shown in Fig. 2, NGHF is optimized by the DDE algorithm. Thus, the convergence of the DDE algorithm is proved. Let $S = \{\mathbf{X}_g \mid g = 1, 2, \ldots, G\}$ be the population sequence, and $\mathbf{x}^*$ be an optimal solution of a binary discrete problem. The convergence of the DDE algorithm can be defined as follows.

***Theorem* 1.** Given the binary searching space of the optimal solution $\mathbf{x}^*$, if $\mathbf{x}^*$ satisfies the following equation:

$$\lim_{G \to \infty} P\{\mathbf{x}^* \in \mathbf{X}_G\} = 1, \tag{15}$$

then the DDE algorithm converges in probability to the optimal solution $\mathbf{x}^*$ of a binary optimization problem.

***Proof of Theorem* 1.** First is to introduce the Markov chain of the DDE algorithm, and then prove its convergence.

1) Markov Chain of DDE

According to prior research [38–41], a discrete Markov chain for DDE is provided.

Initialization. Denote the population at iteration g as follows:

$$\mathbf{X}_g = \{\mathbf{x}_1^g, \mathbf{x}_2^g, \ldots, \mathbf{x}_k^g, \ldots, \mathbf{x}_{NP}^g\},\ k \in \{1, 2, \ldots, NP\} \tag{16}$$

Mutation. After the mutation phase, a new individual was generated based on the different individuals. According to Eqs. (9)-(11), defining the population transition probability of the mutation operation as follows:

$$P_{\mathbf{x}_k^g \to m_k^g} = \varepsilon_k, \quad \left\{0 \le \varepsilon_k \le \frac{1}{(NP-1)(NP-2)(NP-3)}\right\} \tag{17}$$

Crossover. From Eq. (12), the probability of the crossover operation is as follows:

$$P_{(\mathbf{x}_k^g, m_k^g) \to t_k^g} = \gamma_k, \quad (0 \le \gamma_k \le 1) \tag{18}$$

Selection. According to Eq. (13), the probability of the selection operation generating a new individual $\mathbf{x}_k^{g+1}$ is:

$$P_{select} = \begin{cases} 1, & \text{if fitness}(t_k^g \mid \theta) \text{ is better than fitness}(\mathbf{x}_k^g \mid \theta), \\ 0, & \text{otherwise}, \end{cases} \tag{19}$$

Note that the three stages of mutation, crossover, and selection are independent of each other in DDE. Thus, the probability of individual transition from state $\mathbf{x}_k^g$ to $\mathbf{x}_k^{g+1}$ can be denoted as follows:

$$P_{\mathbf{x}_k^g \to \mathbf{x}_k^{g+1}} = P_{\mathbf{x}_k^g \to m_k^g} \cdot P_{(\mathbf{x}_k^g, m_k^g) \to t_k^g} \cdot P_{\text{select}}. \tag{20}$$

***Lemma* 1.** The population sequence $S = \{\mathbf{X}_g \mid g = 1, 2, \ldots, G\}$ is a finite homogeneous Markov chain.

Proof. According to Eqs. (18)-(20), $P(\mathbf{X}_g \to \mathbf{X}_{g+1})$ is only related to $\mathbf{X}_g$. It is proven that the population sequence S has the properties of a Markov chain. Besides, the population sequence $S = \{\mathbf{X}_g \mid g = 1, 2, \ldots, G\}$ is a finite homogeneous Markov chain because: a) the length of the population sequence is discrete and finite with a total G states, and b) the probabilities of mutation, crossover, and selection of the DDE algorithm are independent of g.

2) Convergence Proofs of the DDE Algorithm

Based on Eq. (20), using the DDE algorithm to optimize $\mathbf{X}_g$ can be represented by the following equation:

$$\mathbf{X}_k^{g+1} = \mathbf{X}_k^g \cdot P_{\mathbf{x}_k^g \to m_k^g} \cdot P_{(\mathbf{x}_k^g, m_k^g) \to t_k^g} \cdot P_{\text{select}}, \tag{21}$$

where $k \in \{1, 2, \ldots, NP\}$.

Next, we prove Eq. (21) is convergent as follows.

***Lemma* 2.** In the DDE algorithm, for any two neighboring generations of populations $\mathbf{X}_g$ and $\mathbf{X}_{g+1}$, fitness($\mathbf{X}_{g+1} \mid \theta$) is better than fitness($\mathbf{X}_g \mid \theta$).

Proof. According to Eq. (13), after the selection phase, $\mathbf{x}_k^{g+1}$ always gets better fitness than $\mathbf{x}_k^g$. Based on Eqs. (16)-(20), it is easy to see that fitness($\mathbf{X}_{g+1} \mid \theta$) is better than fitness($\mathbf{X}_g \mid \theta$).

***Theorem* 2.** The population sequence $S = \{\mathbf{X}_g \mid g = 1, 2, \ldots, G\}$ is a finite homogeneous irreducible aperiodic Markov chain.

Proof. Lemma 2 has proven that $S = \{\mathbf{X}_g \mid g = 1, 2, \ldots, G\}$ is a finite homogeneous Markov chain. In computing the probability distribution of the next state of the population in a Markov chain, summing over all the next possible states:

$$\sum (\mathbf{X}_{g+1} \in S) \cdot P\left(\mathbf{X}_g \cdot P_{\mathbf{x}_k^g \to \mathbf{x}_k^{g+1}} = \mathbf{X}_{g+1}\right) = \mathbf{1}. \tag{22}$$

Among Eq. (22), $\mathbf{X}_{g+1} \in S$, any one state $\mathbf{X}_g$ can reach any other state $\mathbf{X}_{g+1}$ with non-zero probability. Thus, the population sequence S is an irreducible Markov chain.

Then, according to Eq. (21), the following inference is achieved:

$$P\left(\mathbf{X}_g \cdot P_{\mathbf{x}_k^g \to \mathbf{x}_k^{g+1}} = \mathbf{X}_{g+1}\right) = \sum S \cdot P_{\mathbf{x}_k^g \to m_k^g} \cdot P_{(\mathbf{x}_k^g, m_k^g) \to t_k^g} \cdot P_{select}. \tag{23}$$

Based on Eqs. (17)-(19), we infer that:

$$P_{\mathbf{x}_k^g \to m_k^g} > 0, P_{(\mathbf{x}_k^g, m_k^g) \to t_k^g} > 0, P_{select} > 0. \tag{24}$$

By Eqs. (23) and (24), the following inference is obtained:

$$P\left(\mathbf{X}_g \cdot P_{\mathbf{x}_k^g \to \mathbf{x}_k^{g+1}} = \mathbf{X}_{g+1}\right) > 0. \tag{25}$$

Eq. (25) proves that the probability of getting state $\mathbf{X}_{g+1}$ after a series of transfers from state $\mathbf{X}_g$ is greater than zero. Then the Markov chain of the DDE algorithm is acyclic.

Theorem 3. Given a global optimal state set C as follows:

$$C = \{\mathbf{X} \mid \text{fitness}(\mathbf{X} \mid \theta) \text{ is better than fitness}(\mathbf{x}^* \mid \theta)\}, \tag{26}$$

where $\mathbf{x}^*$ is an optimal solution of a binary optimization problem. In the DDE algorithm, C is a closed set.

Proof. First, Let $P^l_{\mathbf{X}_a, \mathbf{X}_b}$ be the transfer probability from the state from the state $\mathbf{X}_a$ to $\mathbf{X}_b$ after l steps. The steps for transferring from the state $\mathbf{X}_a$ to $\mathbf{X}_b$ in a population sequence S are as follows:

$$\begin{aligned} P^l_{\mathbf{X}_a, \mathbf{X}_b} = &\sum_{\mathbf{X}_{\varepsilon_1} \in S} \sum_{\mathbf{X}_{\varepsilon_2} \in S} \cdots \sum_{\mathbf{X}_{\varepsilon_{l-1}} \in S} P\big(\mathbf{X}_a \cdot P_{\mathbf{x}_k^a \to \mathbf{x}_k^{\varepsilon_1}} = \mathbf{X}_{\varepsilon_1}\big) \cdot \\ &P\big(\mathbf{X}_{\varepsilon_1} \cdot P_{\mathbf{x}_k^{\varepsilon_1} \to \mathbf{x}_k^{\varepsilon_2}} = \mathbf{X}_{\varepsilon_2}\big) \ldots P\big(\mathbf{X}_{\varepsilon_{l-1}} \cdot P_{\mathbf{x}_k^{\varepsilon_{l-1}} \to \mathbf{x}_k^b} = \mathbf{X}_b\big), \\ &s.t.\ \forall l (l \geq 1), \forall \mathbf{X}_a = \{\mathbf{x}_1^a, \mathbf{x}_2^a, \ldots, \mathbf{x}_{N_P}^a\} \in C, \forall \mathbf{X}_b \notin C. \end{aligned} \tag{27}$$

In Eq. (27), at each step, there exists a transfer from $\mathbf{X}_{t-1} \in C$ to $\mathbf{X}_t$, and $t \in \{1, 2, \ldots, l\}$. There is $\text{fitness}(\mathbf{X}_t|\theta) < \text{fitness}(\mathbf{X}_{t-1}|\theta) = \text{fitness}(\mathbf{x}^*|\theta), s.t.\ \mathbf{X}_t \notin C$. However, according to **Lemma 2**, in each step, $\exists P(\mathbf{X}_{t-1} \to \mathbf{X}_t) = 0$, so based on Eq. (27), $P^l_{\mathbf{X}_a, \mathbf{X}_b} = 0$. Hence, according to the Chapman-Kolmogorov equation [39–41], C is a closed set.

Theorem 4. Based on the population sequence S of the DDE algorithm, there is no non-empty closed set C', *s.t.* $C' \cap C = \emptyset$.

Proof. (Proof by contradiction). Suppose there exists a non-empty closed set C' such that $C' \cap C = \emptyset$, and let $\mathbf{X}_a = \{\mathbf{x^*}^a_1, \mathbf{x^*}^a_2, \ldots, \mathbf{x^*}^a_{N_P}\} \in C, \forall \mathbf{X}_b = \{\mathbf{x}_1^b, \mathbf{x}_2^b, \ldots, \mathbf{x}_{N_P}^b\} \in C'$, there is a $\text{fitness}(\mathbf{x^*}^a_k|\theta)$ that is better than $\text{fitness}(\mathbf{x}_k^b|\theta)$. Then, according to Eq. (26), there exists $P(\mathbf{X}_{\varepsilon_{g+1}} \to \mathbf{X}_{\varepsilon_{g+a+1}}) > 0$, so $P^l_{\mathbf{X}_a, \mathbf{X}_b} > 0$. However, according to $C' \cap C = \emptyset$ defined in the assumption, there should not be the possibility of a l-steps transfer between states $\mathbf{X}_a$ and $\mathbf{X}_b$. Therefore, this assumption is incorrect, which implies that there does not exist a non-empty closed set C'.

Theorem 5 [38, 39]. Given a non-empty closed set E and a state $\mathbf{X}_j$. The transfer probability P_j satisfies the following conditions:

$$\begin{aligned} &\lim_{g \to \infty} P(\mathbf{X}_g = \mathbf{X}_j) = P_j, \quad \text{if } \mathbf{X}_j \in E, \\ &\lim_{g \to \infty} P(\mathbf{X}_g = \mathbf{X}_j) = 0, \quad \text{if } \mathbf{X}_j \notin E, \end{aligned} \tag{28}$$

s.t. $\exists$ a non-empty closed set V satisfy $V \cap E = \emptyset$.

Theorem 6. In the case of infinite iterations, the state $\mathbf{X}_G$ will enter into the global optimal state set C in the DDE algorithm, i.e., $\mathbf{X}_G$ will converge to the optimal solution as follows:

$$\lim_{G \to \infty} \mathbf{X}_G \in C \tag{29}$$

Proof. Combining the results of Theorems 2-5 to conclude that, as the number of iterations tends to infinity, in the DDE algorithm, the state sequence S will converge to set C.

Based on the above inferences in Theorems 2-6, we evidently see that *Theorem* 1 holds. Note that in each training iteration, the hash factor matrices W and Q are generated by the DDE algorithm. Consequently, W and Q converge, indicating that as the algorithm iteratively progresses, each update of $\{W, Q\}$ outperforms its predecessor, resulting in a gradual decrease in the value of the objective function Eq. (7). Therefore, the convergence of the NGHF model is guaranteed.

E. Algorithm Design and Complexity Analysis

First, an algorithm for the DDE algorithm is designed. Algorithm 1 shows the pseudo-code of the DDE algorithm, whose computational cost relies on:

a) Initialization: $T_{11} = \Theta(NP \times D)$, and

b) Training: $T_{12} = \Theta(NP \times G)$.

The DDE algorithm sets G in a small range from 2 to 10. Notably, it is common that $G \ll D \ll NP$. Consequently, the total time complexity of Algorithm 1 comes to:

$$T_1 = T_{11} + T_{12} = \Theta\big(NP \times (D + G)\big) \approx \Theta(NP \times D).$$

Second, based on the DDE algorithm, Algorithm 2 is designed for the NGHF model, whose computational cost relies on:

a) Initialization: $T_{21} = \Theta\big((|U| + |I|) \times D\big)$, and

b) Training: $T_{22} = \Theta\big(t \times (|U| + |I|) \times D\big)$.

So the total time complexity of Algorithm 2 comes to:

$$T_2 = T_{21} + T_{22} \approx \Theta\big(T_{\max} \times (|U| + |I|) \times (D + G)\big).$$

V. Experiments and Results

The next experiments aim to answer the following four research questions:

- **RQ.1**: How do the hyperparameters of the NGHF model impact its prediction performance?
- **RQ.2**: How do different mutation strategies and mutation factor impact the prediction performance of the NGHF model?
- **RQ.3**: How does the proposed NGHF model perform compared with other state-of-the-art hashing models?
- **RQ.4**: How does the proposed NGHF model perform compared with the basic real-valued LF model?

A. General Settings

Datasets. Nine publicly available datasets [14, 38, 42, 51, 52] from various real-world applications are adopted for benchmarking the experiments. They are real HDI datasets generated by industrial applications. TABLE II summarizes their properties. Amazon is a collection of product reviews from the Amazon website. Ciao is a collection of product reviews from the Ciao online shopping platform. Epinion

Algorithm 1 DDE

Input: Λ, G, NP, D, λ
Output: $\mathbf{X}$

/* Initialization */ T_{11}
Initialize $\mathbf{X_1}^{|NP|\times D}$ with $\{+1,-1\}$ randomly
Initialize $g=1$ and maximum-evolution-generation $=G$
Initialize $Fitness=[fitness_1,\ldots,fitness_{NP}]$, $fitness_{\text{best}}=0$
Initialize $SFGSS=8, SFHC=20, Fl=0.1, Fu=0.9, \tau_1=0.1, \tau_2=0.03, \tau_3=0.07, F_k=\text{random}[0,1]$
/* Discrete Differential Evolution */ T_{12}
for $k=1$ to NP **do**
  Calculate $fitness_k$ according to Eq.(14).
end for
for $k=1$ to NP **do**
  Sample a target vector $\mathbf{x}_k^g$ from $\mathbf{X}_g$;
  while not converge and $g \leq G$ **do**
    Randomly select k_1, k_2 from $[1, NP]_t$;
    Calculate F_k according to Eq.(10);
    Calculate m_k^g according to Eqs.(9)–(11);
    Calculate t_k^g according to Eq.(12);
    Calculate $fitness^{\text{new}}$ by t_k^g according to Eq.(14);
    if $fitness_{\text{new}}$ better than $fitness_k$ **then**
      $\mathbf{x}_k^{g+1} = t_k^g$;
    else
      $\mathbf{x}_k^{g+1} = \mathbf{x}_k^g$;
    end if
    $g++$;
  end while
  $k++$;
end for

Algorithm 2 NGHF

Input: Λ, D, $|U|$, $|I|$
Output: W, Q

/* Initialization */ T_{21}
Initialize $W^{|U|\times D}$, $Q^{|I|\times D}$ with $\{+1,-1\}$ randomly;
Initialize hyperparameters λ;
Initialize $t=0$ and maximum-training-round $=T_{\max}$;
/* Training */ T_{22}
while not converge and $t \leq T_{\max}$ **do**
  Update W via Algorithm 1 with fixed Q and $NP=|U|$;
  Update Q via Algorithm 1 with fixed W and $NP=|I|$;
  $t++$;
end while

TABLE II: Properties of all the datasets

Name	$\lvert U\rvert$	$\lvert I\rvert$	$\lvert Z_k\rvert$	Density
Amazon	35,736	38,121	1,960,674	0.14%
Ciao	6,016	47,007	1,459,587	0.52%
Epinion	10,706	8,945	300,303	0.31%
Hetrec-ML	2,113	6,829	841,910	5.83%
MI25M	162,529	24,330	24,890,566	0.63%
Yelp	22,087	14,873	602,518	0.18%
Bitcoin	5,996	6,005	35,592	0.01%
Drug	1,409	1,409	171,223	8.62%
Dating	135,359	168,791	17,359,346	0.08%

TABLE III: Descriptions of all the comparison models.

Model	Description
DCF [26]	It directly learns balanced and uncorrelated hash factors for recommendation tasks by hash collaborative filtering.
DPR [27]	It learns hash factors based on personalized ranking objectives instead of rating prediction objectives in DCF.
CCCF [28]	It utilizes a group of hash factors to depict entities that are subsequently associated with a sparse weight vector consisting of real values.
Neu-hash [29]	A content-aware neural hashing-based collaborative filtering approach. Its version without a content-aware component is chosen for a fair comparison.
VHPHD [30]	It employs projected hamming dissimilarity to improve bit-level importance for enhancing a hashing-based model.
HS-GCN [31]	It is a hamming spatial graph convolutional network for recommendation.
LightFR [48]	It is a federated hashing recommender method for efficient privacy-preserving recommendation.
BGCH+ [50]	It is a bipartite graph contrastive hashing method by utilizing graph convolutional networks to improve Top-N search performance.
MF [13]	The basic real-valued LF model.

is collected by the Epinions.com website. Hetrec-ML stands for a heterogeneous information network data repository for machine learning. MovieLens_25M is a large-scale movie recommendation dataset from the MovieLens website. Yelp is a compilation of user reviews and ratings from the Yelp.com platform. Bitcoin is used to record the trust degree of users in bitcoin transactions. Drug is the drug-drug interaction networks for discovering the potential adverse reactions of combination drugs. Dating is collected from an online dating website LibimSeTi.

Baselines. Comparing the NGHF model with nine related state-of-the-art models, including eight hashing-based models (DCF [26], DPR [27], CCCF [28], Neu-hash [29], VHPHD [30], HS-GCN [31], LightFR [48], and BGCH+ [50]) and one basic real-valued matrix factorization (MF) model [13] MF is utilized as a baseline to show the performance gap between real-valued and binary code models. Their details are summarized in TABLE III.

Implementation Details. DE/Best/1 is selected as the default mutation strategy for the proposed NGHF model. The entities with fewer than 10 entries are removed. Besides, for each user, the items with ratings greater than 3.5 (i.e., ratings of 4 or 5) are selected as positive samples, while others are selected as negative samples. Then, each positive sample with 100 negative samples is paired. In cases where the number of available negative samples is less than 100, the negative samples are packed up together with the respective positive sample and proceed with top-K calculations. By performing the top-K calculation, the rank of the positive sample is determined in the recommendation list. The experimental setup includes five-fold cross-validation. All experiments are conducted on a server with a 2.1 GHz Intel 6230R CPU and 512 GB RAM.

Evaluation metrics. Missing data estimation is a common and significant task to evaluate the performance in representing an HDI matrix [13, 14]. The mean absolute error (MAE) and root mean squared error (RMSE) are widely used as follows:

$$RMSE=\sqrt{\frac{\sum_{(u,i)\in\Gamma}(r_{u,i}-\hat{r}_{u,i})^2}{|\Gamma|}}, MAE=\frac{\sum_{(u,i)\in\Gamma}|r_{u,i}-\hat{r}_{u,i}|_{abs}}{|\Gamma|},$$

where Γ denotes the testing set and $|\cdot|_{abs}$ denotes the absolute value of a given number. Three common ranking evaluation metrics [26–28]: hit rate (HIT), mean reciprocal rank (MRR), and normalized discounted cu-mulative gain (NDCG) are adopted to evaluate the ranking prediction accuracy.

$$HIT@K(u) = \frac{1}{|\Psi_u|}\sum_{j=1}^{|\Psi_u|}\mathbb{I}\big(\mathrm{rank}_{u,j}\le K\big),$$

$$NDCG@K(u)=\frac{DCG@K}{IDCG@K}=\frac{1}{|\Psi_u|}\sum_{j=1}^{|\Psi_u|}\frac{\mathbb{I}\big(\mathrm{rank}_{u,j}\le K\big)}{\log_2\big(\mathrm{rank}_{u,j}+1\big)},$$

$$MRR@K(u) = \frac{1}{|\Psi_u|}\sum_{j=1}^{|\Psi_u|}\frac{\mathbb{I}\big(\mathrm{rank}_{u,j}\le K\big)}{\mathrm{rank}_{u,j}}.$$

where $\mathbb{I}(\cdot)$ denotes the indicator function that returns 1 if the statement is true and 0 otherwise. Ψ_u is a set, which contains all items rated 4 or 5 by the user u. The $\mathrm{rank}_{u,j}$ is the index of the positive sample j in the top-K list.

B. Hyper-parameter Analysis (**RQ.1**)

NGHF has three main hyperparameters, i.e., the regularization parameter λ, the dimension of hash factor space *D*, and the max generation of evolution *G*. Four folds of experiments are conducted *w.r.t.* NGHF: 1) the distributions of LFs with(λ=0.01)/without(λ=0) balance constraints as shown in Fig. 3, 2) the prediction performance with different λ and *D* as shown in Fig 4, 3) the training process with different *G* and *D* as shown in Fig. 5, and 4) the prediction performance with different *G* and *D* as shown in Fig 6. From these results, the following findings are obtained:

- Fig. 3 shows that the balance constraints can shape the distributions of LFs to be more balanced. For example, the sum of the differences in the LFs distributions of 1 and –1 across all bits is 308114 on the dataset Amazon without balance constraints, while that with balance constraints is 203212.
- Figs. 4, 5, and 6 show that a larger *D* makes NGHF have better representation learning ability in most cases, which benefits achieving a higher prediction accuracy.
- Fig. 4 shows that the balance constraints are beneficial for improving the representation learning ability of NGHF at each bit. The balance constraints exhibit consistent performance across a wide range of λ values, as their prediction accuracy shows little sensitivity towards λ.
- Fig. 5 shows that the choice of *G* has a more pronounced impact on the convergence accuracy of NGHF at 8 bit compared with higher bit widths. At 8 bit, NGHF with *G*=1 can converge more deeply than that with a larger *G*, making NGHF achieve a higher convergence accuracy. Moreover, Fig. 5 shows that increasing *G* can accelerate the convergence of NGHF, i.e., *G* can control the convergence speed.
- Fig. 6 shows that the optimal value of *G* is 1 when *D* is small, except for the dataset Drug. The reason may be that the Drug is symmetric, while other datasets are not. Besides, as the number of *D* increases, the effect of *G* diminishes. The reason is that a larger *D* greatly improves the representation learning ability of NGHF, which enables NGHF to converge more deeply. For example, Fig. 5 shows that as *D* increases from 8 bit to 128 bit, the training round with *G*=1 increases from 320 to 2540 on dataset Amazon. Such an gain of representation learning ability reduces the impact of *G*.

C. Mutation Strategies Analysis (**RQ.2**)

*1) **Mutation Strategies**:* To evaluate the influence of different mutation strategies, the six classical mutation strategies [34–38] are compared, i.e., **DE/Rand/1**(**S1**), **DE/Rand/2**(**S2**), **DE/RandToBest/1**(**S3**), **DE/RandToBest/2**(**S4**), **DE/Best/2**(**S5**), and **DE/Best/1** (**S6**). Their mutation strategies are as follows:

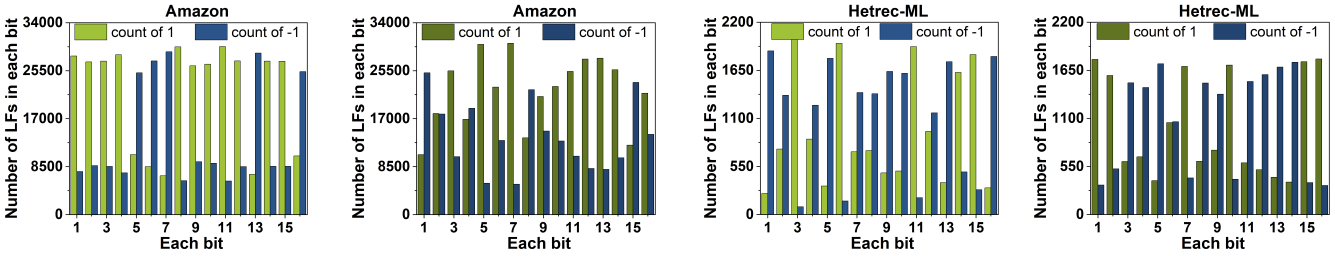
- $DE/Rand/1$: $m_k^g = \mathbf{x}_{r_1}^g + F_k\big(\mathbf{x}_{r_2}^g - \mathbf{x}_{r_3}^g\big)$,
- $DE/Rand/2$: $m_k^g = \mathbf{x}_{r_1}^g + F_k\big(\mathbf{x}_{r_2}^g - \mathbf{x}_{r_3}^g\big) + F_k\big(\mathbf{x}_{r_4}^g - \mathbf{x}_{r_5}^g\big)$,
- $DE/RandToBest/1$: $m_k^g = m_k^g + F\big(\mathbf{x}_{\text{best}}^g - m_k^g\big) + F_k\big(\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g\big)$,
- $DE/RandToBest/2$: $m_k^g = m_k^g + F\big(\mathbf{x}_{\text{best}}^g - m_k^g\big) + F_k\big(\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g\big) + F_k\big(\mathbf{x}_{r_3}^g - \mathbf{x}_{r_4}^g\big)$,
- $DE/Best/2$: $m_k^g = \mathbf{x}_{\text{best}}^g + F_k\big(\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g\big) + F_k\big(\mathbf{x}_{r_3}^g - \mathbf{x}_{r_4}^g\big)$,
- $DE/Best/1$: $m_k^g = \mathbf{x}_{\text{best}}^g + F_k\big(\mathbf{x}_{r_1}^g - \mathbf{x}_{r_2}^g\big)$,

where r_1, r_2, r_3, and r_4 are mutually exclusive integers randomly generated within the range $[1, NP]$ and also different from the index k. The comparison results are shown in Table IV. To better com-pare DE/Best/1(S6) with other variance strategies, we conduct statistical analyses of the loss/win, the Wilcoxon signed rank test, and the Friedman's test following prior studies [43]. From Table IV, we have the following observations:

- DE/Best/1(S6) achieves lower RMSE/MAE than the other strategies on most test cases. It wins 82 cases and only loses eight cases when compared with the other five mutation strategies.
- DE/Best/1(S6) significantly outperforms the other five strategies with high statistical significance (p-value$<$0.05).
- The F-rank value of DE/Best/1(S6) is the lowest among all the variant strategies, indicating its superior performance.

Therefore, these observations provide strong evidence for the superiority of DE/Best/1 over the alternative variants.

*2) **Mutation Factor**:* To evaluate the effectiveness of Eq.(10) for adaptively setting mutation factor F_k, the performance with different fixed mutation factors F_k is tested on all the datasets. The testing results are shown in Table V, where we find that F_k in the range of 0.5-0.9 has relatively better performance than that of 0.1-0.3. For example, F_k with fixed



(a) λ=0,Differences=308114 (b) λ=0.01,Differences=203212 (c) λ=0,Differences=18954 (d) λ=0.01,Differences=17144

Fig. 3: The distribution histogram of LFs of NGHF on the datasets of Amazon and Hetrec-ML with λ=0, 0.01, D=16, and G=1. Note that 'Differences' denotes that the sum of the differences in the LFs distributions of 1 and –1 across all bits.

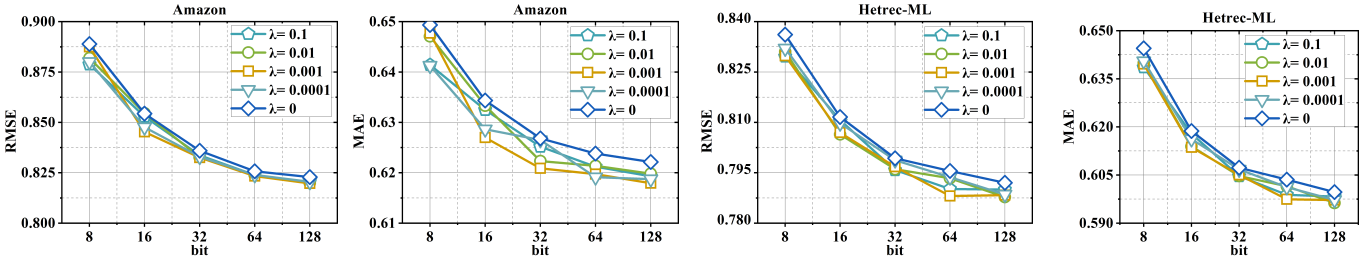
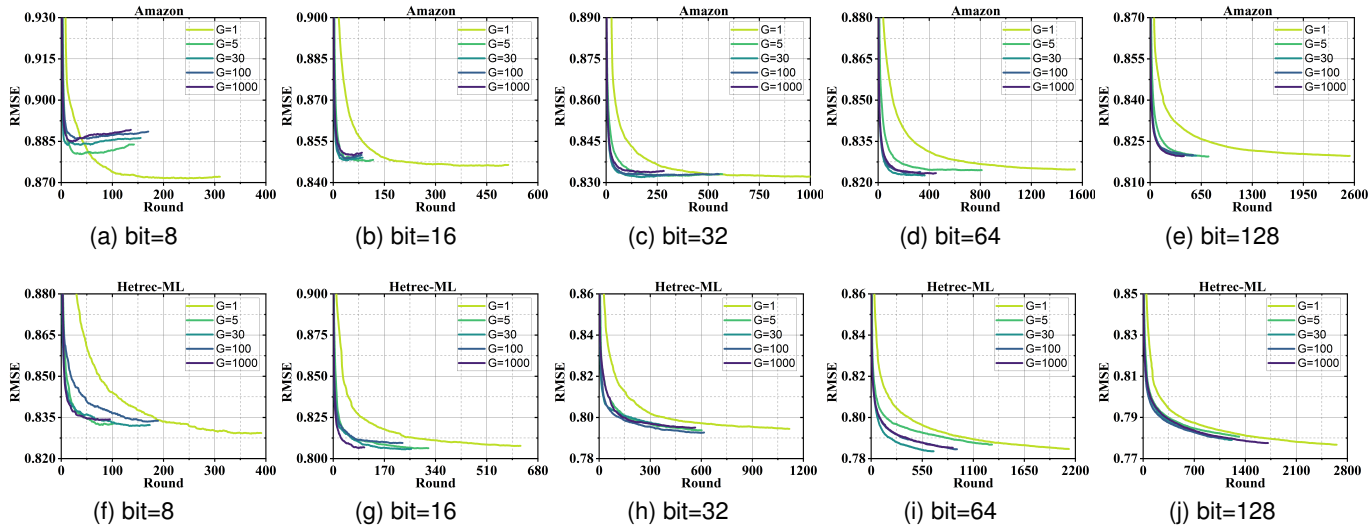


Fig. 4: The prediction performance of NGHF on the datasets of Amazon and Hetrec-ML with different λ=10-5, 10-4, 0.001, 0.01, 0.1, D=8, 16, 32, 64, 128, and G=1.



(a) bit=8 (b) bit=16 (c) bit=32 (d) bit=64 (e) bit=128

(f) bit=8 (g) bit=16 (h) bit=32 (i) bit=64 (j) bit=128

Fig. 5: The training process of NGHF on the datasets of Amazon and Hetrec-ML with G=1, 5, 30, 100, 1000 and D=8, 16, 32, 64, 128.

0.5, 0.7, and 0.9 outperforms adaptive F_k in five, five, and four cases, respectively. While F_k with fixed 0.1 and 0.3 losses all the cases when comparing with adaptive F_k. The reason may be that a small F_k is hard to mutate a diverse individual. In addition, the results of Table V show that different datasets are suitable for the different fixed F_k, which denotes that fixed F_k needs fine-tuning for different datasets. In comparison, although adaptive F_k following Eq.(10) does not have the best performance in most cases, it still significantly outperforms each fixed F_k and achieves the best performance among all the settings with fixed F_k.

D. Comparison with State-of-the-arts Models (**RQ.3**)

1) ***Prediction Accuracy Comparison****:* NGHF is compared with eight state-of-the-art hashing-based models as introduced in Table III. To make a fair comparison, the dimension of the hash factor space D is set as $\{32, 64\}$ for all the models. The detailed comparison results are presented in Tables VI and VII. The statistical analyses are also done on these results following prior studies [43]. From these results, the following notable observations can be obtained:

- Table VI shows that NGHF exhibits much higher accuracy of missing data estimation than the eight hash-based

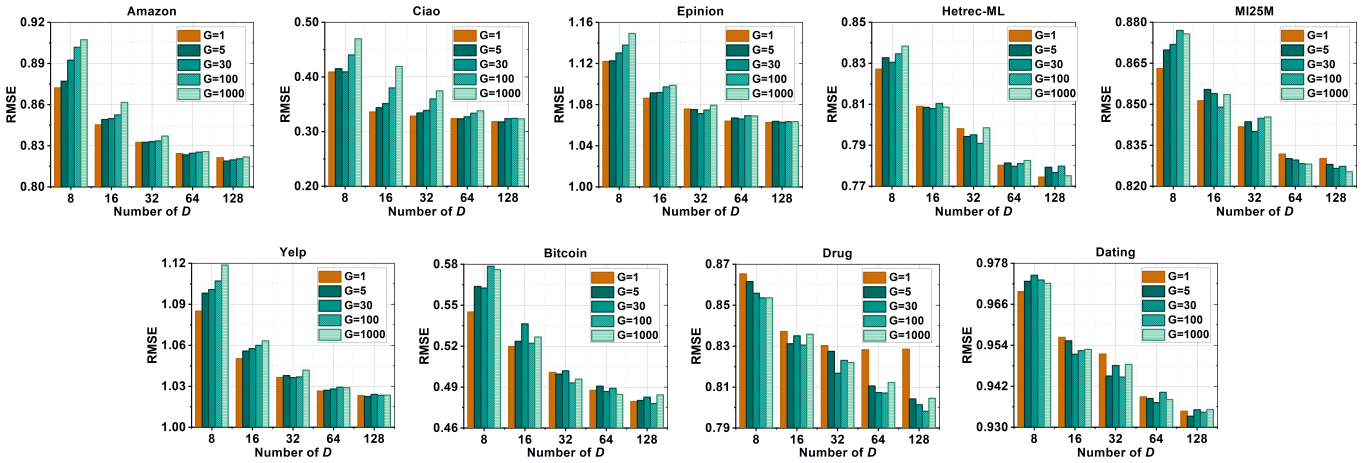


Fig. 6: The prediction performance of NGHF on all the datasets with G= 1, 5, 30, 100, 1000 and D=8, 16, 32, 64, 128.

TABLE IV: Comparison results with different Mutation Strategies on all the datasets, where G=5, D=16, and λ=0.01.

Dataset	Metric	Mutation Strategy					
		S1	S2	S3	S4	S5	S6
Amazon	MAE	0.6424	0.6423	0.6243•	0.6352	0.6292	0.6269
	RMSE	0.8671	0.8675	0.8461	0.8557	0.8509	**0.8453**
Ciao	MAE	0.3146	0.3348	0.2216•	0.2699	0.2217•	0.2304
	RMSE	0.4241	0.4373	0.3429	0.3825	0.3445	**0.3344**
Epinion	MAE	0.8485	0.8493	0.8523	0.8489	0.8531	**0.8422**
	RMSE	1.1004	1.1003	1.0945	1.0942	1.0958	**1.0874**
Hetrec-ML	MAE	0.6401	0.6424	0.619	0.6244	0.6179	**0.6158**
	RMSE	0.8357	0.8372	0.8145	0.8184	0.8100	**0.8050**
MI25M	MAE	0.6683	0.6667	0.6529	0.6548	0.6464•	0.6485
	RMSE	0.8701	0.8686	0.8539	0.8554	0.8515	**0.8505**
Yelp	MAE	0.8369	0.8368	0.8251	0.8324	0.8304	**0.8236**
	RMSE	1.0671	1.0669	1.05	1.0597	1.0596	**1.0491**
Bitcoin	MAE	0.4192	0.4283	0.3732	0.4082	0.3960	**0.3583**
	RMSE	0.5676	0.5794	0.5266	0.5574	0.5446	**0.5191**
Drug	MAE	0.5803•	0.5857•	0.5944	0.5985	0.5987	0.5932
	RMSE	0.8209•	0.8221•	0.8316	0.8315	0.8368	0.8308
Dating	MAE	0.7092	0.7124	0.6911	0.6977	0.6943	**0.6869**
	RMSE	0.9773	0.9723	0.9567	0.9628	0.9611	**0.9513**
Statistical Analysis	loss/win •	2/16	2/16	2/16	0/18	2/16	**8/82**
	p-value ⋆	0.0002	0.0002	0.0027	0.0001	0.0011	-
	F-rank*	4.7778	5.0556	2.6111	3.8889	3.2222	**1.4444**

• The cases of S6 lose the comparison. ⋆NGHF significantly outperforms the comparison model if the p-value$<$0.05. *A lower value denotes a better performance.

comparison models. It has the lower RMSE/MAE values in all the test cases with 16 bits. Moreover, there are only two cases that NGHF has higher RMSE/MAE values than DCF with 64 bits. The overall loss/win scenarios with 16 and 64 bits are 0/144 and 2/142, respectively.

- Table VII shows that NGHF also achieves much higher ranking prediction accuracy in most test cases. The overall loss/win scenarios with 16 and 64 bits are 8/208 and 9/207, respectively. We note that BGCH+ outperforms NGHF on the dataset Drug. The reason may be that Drug has the symmetric networks. BGCH+'s graph contrastive hashing method is suitable for improving symmetric ranking search performance.
- NGHF significantly outperforms the eight hash-based comparison models in both missing data estimation and ranking prediction according to the Wilcoxon signed rank test with the p-values much less than 0.05.

TABLE V: COMPARISON RESULTS BETWEEN FIXED AND ADAPTIVE MUTATION FACTOR F_k ON ALL THE DATASETS, WHERE G=5, D=16, AND λ=0.01.

Dataset	Metric	Mutation factor F_k					
		F_k=0.1	F_k=0.3	F_k=0.5	F_k=0.7	F_k=0.9	F_k=Eq.(10)
Amazon	MAE	0.7179	0.7884	0.6331	0.6274	0.6332	**0.6269**
	RMSE	0.9051	0.9559	0.8429•	0.8516	0.8573	0.8453
Ciao	MAE	0.3194	0.2707	0.2259•	0.2262•	0.2294•	0.2304
	RMSE	0.4472	0.4273	0.3356	0.3483	0.3409	**0.3344**
Epinion	MAE	0.9822	0.9616	0.8369•	0.8447	0.8459	0.8422
	RMSE	1.1984	1.1849	1.0881	1.0958	1.0970	**1.0874**
Hetrec-ML	MAE	0.6919	0.7149	0.6117•	0.6112•	0.6164	0.6158
	RMSE	0.8973	0.9286	0.8056	0.8052	0.8076	**0.8050**
MI25M	MAE	0.7392	0.7313	0.6529	0.6490	0.6525	**0.6485**
	RMSE	0.9767	0.9413	0.8549	0.8509	0.8534	**0.8505**
Yelp	MAE	0.8736	0.8403	0.8207•	0.8273	0.8271	0.8236
	RMSE	1.0897	1.0698	1.0504	1.0549	1.0572	**1.0491**
Bitcoin	MAE	0.5795	0.4310	0.3593	0.3545•	0.3788	0.3583
	RMSE	0.7520	0.5812	0.5229	0.5187•	0.5370	0.5191
Drug	MAE	1.0448	1.0484	0.6962	0.5904•	0.5846•	0.5932
	RMSE	1.3050	1.2973	0.9190	0.8326	0.8263•	0.8308
Dating	MAE	1.1492	0.9477	0.7085	0.6913	0.6785•	0.6869
	RMSE	1.4174	1.1886	0.9911	0.9582	0.9537	**0.9513**
Statistical Analysis	loss/win•	0/18	0/18	5/13	5/13	4/14	**14/76**
	p-value ⋆	0.0001	0.0001	0.0490	0.0388	0.0203	-
	F-rank*	5.7222	5.2778	2.6667	2.4444	3.1111	**1.7778**

• The cases of S6 lose the comparison. ⋆NGHF significantly outperforms the comparison model if the p-value$<$0.05. *A lower value denotes a better performance.

- NGHF can significantly decrease both memory usage and processing time compared to the hash-based models when learning short binary representations in Hamming Space. For example, Tables VI and VII shows that NGHF with only 16 bits achieves a lower F-rank value than other hash-based models with 64 bits in Friedman's test.

In summary, these observations demonstrate that NGHF achieves significantly superior performance over other hash-based learning baselines.

*2) **Computational Efficiency Comparison**:* To evaluate the computational efficiency of all the hash-based comparison models and NGHF, an $M\times N$ matrix is constructed by randomly sampling from a normal distribution. By following [27], M is fixed at 100 and N is incrementally increased to simulate dataset growth. Fig. 7 records all the involved models' CPU running time for once training. It can be seen that all the models have a similar CPU running time on the small

TABLE VI: THE COMPARISON RESULTS ON PREDICTION OF RATING ACCURACY.

Dataset	Metric	DCF		DPR		CCCF		Neu-hash		VHPHD		HS-GCN		LightFR		BGCH+		NGHF	
		16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit
Amazon	MAE	0.6678	0.6717	1.0337	1.0185	1.0160	0.9397	0.7323	0.6801	0.9845	0.9980	0.9462	1.3361#	1.4102	1.4230	0.7249	0.6908	0.6269	0.6214
	RMSE	0.8738	0.8562	1.2648	1.2235	1.2242	1.1902	0.9411	0.8571	1.1765	1.1562	1.2382	1.5195#	1.5694	1.5842	0.9836	0.9377	0.8453	0.8241
Ciao	MAE	0.2535	0.2492	0.6471	0.5980	0.8644	0.4475	0.8206	0.3232	0.6401	0.5939	1.0630	1.0804#	1.0059	0.9675	0.5436	0.5140	0.2304	0.2145
	RMSE	0.3984	0.3595	0.8436	0.7797	1.0329	0.6631	0.9175	0.4054	0.8323	0.7758	1.3323	1.2215#	1.1600	1.0535	0.6888	0.6455	0.3344	0.3268
Epinion	MAE	0.9055	0.9162	1.4850	1.4890	1.4055	1.4041	0.8980	0.8934	1.2190	1.2384	1.5455	1.5070	1.4433	1.4178	1.0175	0.8987	0.8422	0.8351
	RMSE	1.1297	1.1103	1.7336	1.7079	1.6170	1.6018	1.1273	1.0975	1.4196	1.3958	1.7793	1.6816	1.6361	1.5762	1.3108	1.1860	1.0874	1.0669
Hetrec-ML	MAE	0.6442	0.6099	0.9495	0.9347	1.2137	0.9271	0.8920	0.7461	1.1648	1.0435	1.1477	0.9744	0.9682	0.8918	0.7646	0.7422	0.6158	0.5931
	RMSE	0.8369	0.7926	1.2129	1.1875	1.2342	1.1380	1.1186	0.9599	1.3859	1.2270	1.4404	1.1662	1.1901	1.1045	0.9750	0.9535	0.8050	0.7860
Ml25M	MAE	0.6703	0.6392	0.9500	0.9205	0.8889	0.7369	1.3587	0.6634	1.0711	0.8806	1.3592#	1.5686#	1.0593	0.9968	0.8221	0.7452	0.6485	0.6311
	RMSE	0.8696	0.8297●	1.2125	1.1782	1.0958	0.9267	1.6085	0.8367	1.2918	1.1226	1.7027#	1.1489#	1.2879	1.2046	1.0603	0.9603	0.8505	0.8298
Yelp	MAE	0.8915	0.8843	1.0495	1.0197	1.2650	1.3414	0.8877	0.8576	0.8898	0.8468	1.0630#	1.1568#	1.2385	1.1981	0.9257	0.9074	0.8236	0.8074
	RMSE	1.1107	1.0836	1.3097	1.2641	1.4943	1.5526	1.1191	1.0724	1.1891	1.1177	1.3324#	1.3596#	1.4523	1.3863	1.1597	1.1357	1.0491	1.0283
Bitcoin	MAE	0.5865	0.4031	0.8233	0.7465	0.7026	0.5582	0.7854	0.7125	0.8457	0.8272	0.6390	0.4702	0.7003	0.5542	0.5067	0.4970	0.3583	0.3272
	RMSE	0.7668	0.5814	1.0996	1.0078	0.9061	0.7558	0.9847	0.8475	1.1112	1.0467	0.8359	0.6654	0.9103	0.7539	0.6813	0.6670	0.5191	0.4868
Drug	MAE	0.6362	0.5991	1.3346	1.2975	0.8028	0.9447	0.6547	0.6321	0.8285	0.7009	1.3097	1.3225	1.3019	1.2438	1.0288	0.9399	0.5932	0.5846
	RMSE	0.8553	0.7996●	1.5839	1.4993	1.0196	1.1210	0.8754	0.8568	1.0468	0.9628	1.5655	1.4877	1.5063	1.4051	1.2364	1.1623	0.8308	0.8072
Dating	MAE	0.7352	0.7186	1.4366	1.2119	1.2259	1.1200	0.9547	0.9024	1.0394	0.9872	1.6996#	1.6015#	1.4777	1.4025#	1.0671	0.9751	0.6869	0.6711
	RMSE	0.9765	0.9496	1.6930	1.4511	1.4634	1.3453	1.1457	1.0847	1.2985	1.2238	2.1266#	2.0874#	1.7394	1.6842#	1.3855	1.3284	0.9513	0.9384
Statistical Analysis	loss/win ●	0/18	2/16	0/18	0/18	0/18	0/18	0/18	0/18	0/18	0/18	0/18	0/18	0/18	0/18	0/18	0/18	**0/144**	**2/142**
	p-value ⋆	0.0001	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	-	-
	F-rank*	5.2222	3.0556	14.5000	12.6667	12.8889	10.1111	9.2778	5.2222	11.9444	9.8889	15.5000	14.1111	14.8889	12.7778	8.5000	6.9444	**2.3889**	**1.1111**

●The cases of NGHF lose the comparison. ⋆NGHF significantly outperforms the comparison model if the *p*-value is lower than 0.05. *A lower value denotes a better performance. # Based on the same experimental setup, the HS-GCN model failed to converge to a stable value within seven days on a certain dataset.

TABLE VII: THE COMPARISON RESULTS OF RANKING PREDICTION ACCURACY.

Dataset	Metric	DCF		DPR		CCCF		Neu-hash		VHPHD		HS-GCN		LightFR		BGCH+		NGHF	
		16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit	16bit	64bit
Amazon	HIT	0.9881	0.9890	0.9126	0.9137	0.9669	0.9675	0.9280	0.9896	0.9864	0.9874	0.9816	0.9817#	0.9812	0.9806	0.9860	0.9880	0.9897	0.9906
	MRR	0.6526	0.6774	0.4601	0.4706	0.5662	0.5746	0.5015	0.6842	0.6260	0.6458	0.5848	0.5877#	0.5726	0.5728	0.6242	0.6597	0.6750	0.6998
	NDCG	0.7363	0.7551	0.5691	0.5774	0.6646	0.6710	0.6182	0.7605	0.7158	0.7311	0.6832	0.6857#	0.6741	0.6742	0.7143	0.7417	0.7520	0.7725
Ciao	HIT	0.9870	0.9928	0.9058	0.9225	0.9036	0.9446	0.9718	0.9879	0.9367	0.9831	0.9263	0.9371#	0.8984	0.8988	0.9711	0.9704	0.9945	0.9958
	MRR	0.8221	0.8697	0.5465	0.6014	0.5418	0.6327	0.7258	0.8314	0.6089	0.8076	0.5674	0.6116#	0.4920	0.4934	0.7522	0.7537	0.8735	0.8949
	NDCG	0.8634	0.9006	0.6342	0.6802	0.6302	0.7083	0.7871	0.8629	0.6897	0.8513	0.6559	0.6920#	0.5915	0.5927	0.8090	0.8099	0.9038	0.9200
Epinion	HIT	0.9973	0.9978	0.9842	0.9841	0.9897	0.9892	0.9975	0.9980	0.9969	0.9975	0.9923	0.9930	0.9935	0.9935	0.9948	0.9972	0.9979	0.9981
	MRR	0.7809	0.8015	0.5604	0.5564	0.6502	0.6571	0.7767	0.7956	0.7626	0.7946	0.6102	0.6172	0.6307	0.6293	0.6558	0.7725	0.8027	0.8160
	NDCG	0.8359	0.8515	0.6657	0.6627	0.7350	0.7401	0.8329	0.8472	0.8222	0.8463	0.7065	0.7119	0.7224	0.7211	0.7416	0.8296	0.8531	0.8626
Hetrec-ML	HIT	0.6709	0.7141	0.4538	0.4761	0.5054	0.4811	0.5450	0.5700	0.5796	0.6300	0.5505	0.5728	0.5014	0.4938	0.6021	0.6287	0.6852	0.7221
	MRR	0.3086	0.3550	0.1725	0.1906	0.2071	0.1823	0.2157	0.2390	0.3077	0.3294	0.2265	0.2539	0.1923	0.1853	0.2839	0.3015	0.3311	0.3799
	NDCG	0.3941	0.4399	0.2382	0.2573	0.2770	0.2520	0.2930	0.3220	0.3921	0.4091	0.3025	0.3290	0.2648	0.2577	0.3590	0.3788	0.4129	0.4598
Ml25M	HIT	0.8845	0.9058●	0.7335	0.7424	0.7983	0.8439	0.7229	0.8576	0.8597	0.8768	0.8131#	0.8344#	0.8059	0.8051	0.8761	0.8924	0.8901	0.8960
	MRR	0.4878	0.5441	0.3033	0.3147	0.3905	0.4490	0.2926	0.4727	0.4590	0.4924	0.3724#	0.4046#	0.3661	0.3646	0.4976	0.5299	0.5059	0.5527
	NDCG	0.5837	0.6319●	0.4049	0.4158	0.4800	0.5437	0.3942	0.5651	0.4704	0.5860	0.4779#	0.5078#	0.4714	0.4701	0.5889	0.6176●	0.5988	0.6130
Yelp	HIT	0.9941	0.9947	0.9798	0.9804	0.9817	0.9815	0.9939	0.9944	0.9929	0.9940	0.9909#	0.9910#	0.9900	0.9899	0.9935	0.9938	0.9948	0.9959
	MRR	0.6993	0.7204	0.5728	0.5798	0.5884	0.5850	0.6946	0.7143	0.6931	0.7223	0.6352#	0.6326#	0.6176	0.6204	0.6956	0.7107	0.7249	0.7513
	NDCG	0.7738	0.7898	0.6738	0.6792	0.6861	0.6835	0.7702	0.7851	0.7688	0.7909	0.7247#	0.7225#	0.7113	0.7134	0.7709	0.7822	0.7931	0.8127
Bitcoin	HIT	0.9371	0.9521	0.9662●	0.9633	0.9232	0.9275	0.9347	0.9412	0.9585●	0.9687	0.9322	0.9451	0.9216	0.9318	0.9112	0.9289	0.9571	0.9725
	MRR	0.4737	0.4756	0.5176	0.5385	0.4730	0.4771	0.4811	0.5065	0.5027	0.5410	0.4834	0.4864	0.4567	0.4529	0.5350●	0.5654●	0.5188	0.5542
	NDCG	0.5844	0.5901	0.6383●	0.6438	0.5832	0.5853	0.5991	0.6213	0.6146	0.6468	0.5921	0.5981	0.5687	0.5688	0.6252	0.6540	0.6257	0.6559
Drug	HIT	0.6887	0.7103●	0.5005	0.5001	0.7791●	0.5980	0.6679	0.6891	0.6698	0.6818	0.4981	0.5131	0.5105	0.5102	0.8064●	0.8475●	0.6911	0.7093
	MRR	0.3218	0.3346	0.1944	0.1916	0.3087	0.2552	0.2865	0.3098	0.2938	0.3132	0.1933	0.1855	0.1861	0.1853	0.4540●	0.5110●	0.3265	0.3371
	NDCG	0.4081	0.4231	0.2654	0.2640	0.3968	0.3361	0.3912	0.4123	0.3832	0.4007	0.2648	0.2623	0.2622	0.2615	0.5381●	0.5918●	0.4132	0.4244
Dating	HIT	0.9545	0.9622	0.7897	0.8210	0.8123	0.9274	0.8531	0.8654	0.8351	0.8389	0.7839#	0.8023#	0.7855	0.8153#	0.9338	0.9488	0.9597	0.9642
	MRR	0.7303	0.7579	0.3177	0.4049	0.4028	0.7007	0.4355	0.4427	0.4215	0.4291	0.3179#	0.3321#	0.3174	0.3481#	0.6249	0.7157	0.7566	0.7591
	NDCG	0.7869	0.8107●	0.4292	0.5932	0.5863	0.7557	0.5612	0.5774	0.5279	0.5274	0.4281#	0.4412#	0.4281	0.4512#	0.7001	0.7726	0.8004	0.8078
Statistical Analysis	loss/win ●	0/27	4/23	2/25	0/27	1/26	0/27	0/27	0/27	1/26	0/27	0/27	0/27	0/27	0/27	4/23	5/22	**8/208**	**9/207**
	p-value ⋆	0.0001	0.0002	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0043	0.0090	-	-
	F-rank*	6.4815	3.7407	15.3704	14.1852	13.2963	12.2593	11.1667	6.3333	9.4815	6.0185	13.2037	11.8519	15.0000	14.7593	7.5185	5.4074	**3.4444**	**1.4815**

●The cases of NGHF lose the comparison. ⋆NGHF significantly outperforms the comparison model if the *p*-value is lower than 0.05. *A lower value denotes a better performance. # Based on the same experimental setup, the HS-GCN model failed to converge to a stable value within seven days on a certain dataset.

dataset, except for HS-GCN. As the dataset size increases, the computational costs of all models experience rapid growth. In general, DCF has a relatively higher computational efficiency than other models on middle and large datasets, HS-GCN has the worst performance on all dataset sizes, and NGHF has comparable and acceptable computational efficiency. Notably, according to [16], the two latent factor matrices of an LF model could be trained in parallelly by alternating stochastic gradient descent. Following this principle, the two hash factor matrices of NGHF could also be optimized in parallel by designing a parallel DDE algorithm, which can greatly improve NGHF's computational efficiency.

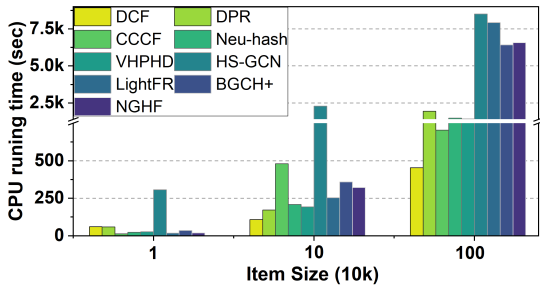


Fig. 7: The comparison of the CPU running time of all the involved models.

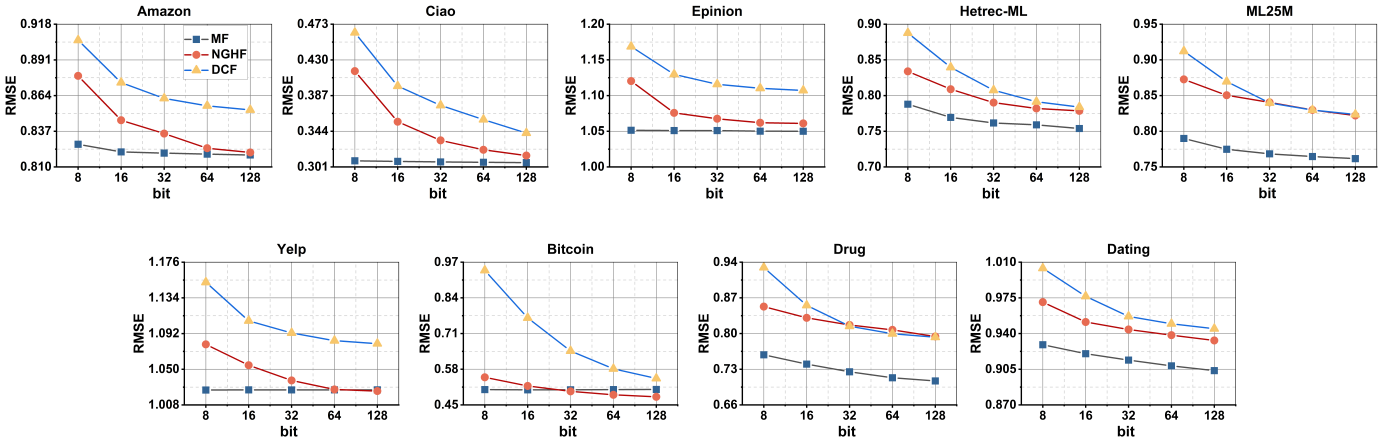


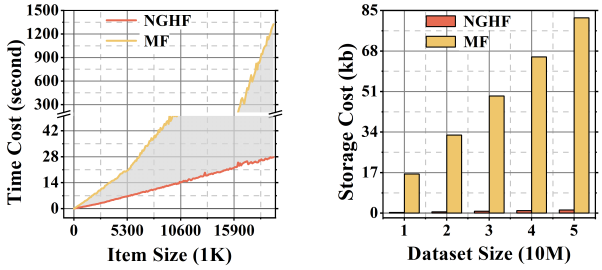
Fig. 8: The RMSE prediction performance of real-valued MF and NGHF in different datasets, where D=8, 16, 32, 64, 128.

*E. Comparison with Basic Real-valued MF Model (**RQ.4**)*

1) Prediction Accuracy Comparison: This set of experiments selects MF, DCF (the best-performing hash-based comparison model), and NGHF to evaluate the prediction performance between real-valued and binary code models. The comparison results on all the datasets are shown in Fig. 8. At the initial stage with fewer binary bits, the prediction accuracy of real-valued MF is much higher than that of hash-based DCF and NGHF due to the information loss during the discretization process. However, as the binary bits increase, the prediction accuracy of DCF and NGHF keeps improving while that of MF improves slightly, making the accuracy gap between real-valued MF and hash-based DCF and NGHF gradually diminish. Finally, NGHF exhibits similar prediction accuracies to MF with 128 bits on specific datasets. The reason is that increasing the number of bits enhances the information representation ability of hash bits for DCF and NGHF. In contrast, although more bits can improve the information representation ability of real-valued MF, such improvement is limited by overfitting because a small number of real-valued bits can already provide strong information representation ability. Especially, the hash-based NGHF even achieves a higher accuracy than the real-valued MF on the dataset Bitcoin. Notably, overfitting also affects the hash-based DCF and NGHF models, as their accuracy gains gradually decrease with an increasing number of binary bits. Besides, NGHF consistently achieves higher accuracy than DCF. Hence, these results demonstrate that the proposed DDE algorithm used in NGHF has a significant effect in avoiding quantization loss of hashing.

2) Time and Storage Costs Comparison: To analyze the difference in computational and storage costs between hash-based NGHF and real-valued MF in the reasoning process as the size of the dataset grows, an $M \times N$ matrix is still constructed by following [27]. M is fixed at 100 and N is incrementally increased to simulate dataset growth. In the real-valued MF, each entity is represented as a real-valued LF vector. Whereas in NGHF, each entity is represented as a binary code in Hamming space.

The results in Figure 9 show that with increasing dataset size, both the computational and storage costs of real-valued MF experience rapid growth. In contrast, that of NGHF increases at a much slower rate, which can be attributed to the following two reasons: (1) in Hamming space, reasoning can be accelerated through bitwise operations (e.g., XOR) instead of real-valued computation, reducing the time required for preference inference; (2) in real-valued space, each dimension of the hidden eigenvector takes up 64 bits, whereas each dimension only requires 1 bit in Hamming space. This significantly reduces the needed storage space during model computation and allows for faster data computation directly in memory. Note that the computational cost of NGHF is still much lower than that of MF, even though NGHF has more binary bits than MF.



(a) Time Cost (b) Storage Cost

Fig. 9: The comparison results of computational and storage costs on the artificial datasets. (a): Computational cost. (b): Storage cost.

F. Summary of Experiments

Based on the above experimental results and analyses on nine real HDI datasets, we draw the following conclusions:

- a) Compared to eight state-of-the-art hash baselines, NGHF achieves significantly higher prediction accuracy for both missing data estimation and ranking prediction.
- b) NGHF demonstrates significant improvements in computational and storage costs compared to the real-valued MF model.

- c) The most suitable mutation strategy for the NGHF model is DE/Best/1, wherein the adaptive mutation factor setting following Eq.(10) is effective.
- d) The dimension of the hash factor is a crucial factor influencing the performance of NGHF. As the dimension of the hash factor increases, the precision of NGHF improves continuously. However, beyond a certain point, the rate of improvement in precision slows down.
- e) The balance constraints are beneficial for improving the representation learning ability of NGHF.
- f) A larger maximum generation of the DDE algorithm could accelerate the convergence speed of NGHF.

VI. Conclusions

In this paper, a discrete differential evolution (DDE) algorithm is proposed to simulate continuous optimization via disabling bits of binary codes based on the projected Hamming dissimilarity. Subsequently, a non-gradient hash factor (NGHF) model is proposed based on the DDE algorithm. To evaluate the proposed NGHF model, extensive experiments are conducted on nine widely used high-dimensional and incomplete (HDI) datasets. The results demonstrate that NGHF outperforms eight state-of-the-art hash learning models. Besides, compared to real-valued models, such as MF [13], NGHF not only achieves amazingly comparable prediction accuracy but also has a significantly higher computational efficiency of reasoning. These results highlight the high accuracy and fast reasoning of NGHF in extracting hidden information from HDI data, which is critical for industrial applications. In the future, we will optimize the two hash factor matrices of NGHF in parallel by designing a parallel DDE algorithm, which can greatly improve NGHF's computational efficiency of training. Besides, we plan to explore a more complex hash-learning scenario for dynamic HDI data based on the Canonical Polyadic tensor factorization [53, 54].

References

[1] M. C. Tsakiris, "Low-Rank Matrix Completion Theory via Plücker Coordinates," IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 8, pp. 10084-10099, 2023.

[2] Y. Xu, E. Wang, Y. Yang and H. Xiong, "GS-RS: A Generative Approach for Alleviating Cold Start and Filter Bubbles in Recommender Systems," IEEE Trans. Knowl. Data Eng., vol. 36, no. 2, pp. 668-681, 2024

[3] N. Zhou, K. -S. Choi, B. Chen, Y. Du, J. Liu and Y. Xu, "Correntropy-Based Low-Rank Matrix Factorization With Constraint Graph Learning for Image Clustering," IEEE Trans. Neural Networks Learn. Syst., vol. 34, no. 12, pp. 10433-10446, 2023.

[4] X. Su et al., "Biomedical Knowledge Graph Embedding With Capsule Network for Multi-Label Drug-Drug Interaction Prediction," IEEE Trans. Knowl. Data Eng., vol. 35, no. 6, pp. 5640-5651, 2023.

[5] T. Cao, Q. Xu, Z. Yang and Q. Huang, "Mitigating Confounding Bias in Practical Recommender Systems With Partially Inaccessible Exposure Status," IEEE Trans. Pattern Anal. Mach. Intell., vol. 46, no. 2, pp. 957-974, Feb. 2024.

[6] C. Wu, D. Lian, Y. Ge, Z. Zhu and E. Chen, "Influence-Driven Data Poisoning for Robust Recommender Systems," IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 10, pp. 11915-11931, Oct. 2023.

[7] S. Zhang et al., "Personalized Latent Structure Learning for Recommendation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 45, no. 8, pp. 10285-10299, 2023.

[8] X. Su, M. Zhang, Y. Liang, Z. Cai, L. Guo, and Z. Ding, "A Tensor-Based Approach for the QoS Evaluation in Service-Oriented Environments," IEEE Trans. Netw. Serv. Manag., vol. 18, no. 3, pp. 3843-3857, 2021.

[9] Y. Lin, P. Ren, Z. Chen, Z. Ren, D. Yu, J. Ma, M. d. Rijke, and X. Cheng, "Meta Matrix Factorization for Federated Rating Predictions," In Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2020, pp. 981–990.

[10] D. Wu, Z. Li, Z. Yu, Y. He, and X. Luo, "Robust Low-Rank Latent Feature Analysis for Spatiotemporal Signal Recovery," IEEE Trans. Neural Networks Learn. Syst., vol.36, no.2, pp.2829-2842, 2025.

[11] M. Li and Y. Song, "An Improved Non-Negative Latent Factor Model for Missing Data Estimation via Extragradient-Based Alternating Direction Method," IEEE Trans. Neural Networks Learn. Syst., vol. 34, no. 9, pp. 5640-5653, 2023.

[12] C. Leng, H. Zhang, G. Cai, I. Cheng, and A. Basu, "Graph regularized Lp smooth non-negative matrix factorization for data representation," IEEE CAA J. Autom. Sinica 6., vol. 6, no. 2, pp. 584-595, 2019.

[13] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," Computer, vol. 42, no. 8, pp. 30-37, 2009.

[14] D. Wu, P. Zhang, Y. He and X. Luo, "MMLF: Multi-Metric Latent Feature Analysis for High-Dimensional and Incomplete Data," IEEE Trans. Services Comput., vol. 17, no. 2, pp. 575-588, 2024.

[15] L. Lv, P. Hu, D. Bardou, Z. Zheng and T. Zhang, "Community Detection in Multilayer Networks Via Semi-Supervised Joint Symmetric Nonnegative Matrix Factorization, " IEEE Trans. Netw. Sci. Eng. vol. 10, no. 3, pp. 1623-1635, 2023.

[16] D. Wu, X. Luo, Y. He, and M. Zhou, "A Prediction-sampling-based Multilayer-structured Latent Factor Model for Accurate Representation to High-dimensional and Sparse Data," IEEE Trans. Neural Networks Learn. Syst., vol. 35, no. 3, pp. 3845-3858, 2024.

[17] K. Zhou and H. Zha, "Learning binary codes for collaborative filtering," in Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, in KDD 2012. 2012, pp. 498–506.

[18] L. Wang, Y. Pan, C. Liu, H. Lai, J. Yin, and Ye Liu; "Deep Hashing With Minimal-Distance-Separated Hash Centers", In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023, pp. 23455-23464.

[19] J. Wang, T. Zhang, j. song, N. Sebe and H. T. Shen, "A Survey on Learning to Hash," IEEE Trans. Pattern Anal. Mach. Intell., vol. 40, no. 4, pp. 769-790, 2018.

[20] X.-S. Wei, Y. Shen, X. Sun, P. Wang and Y. Peng, "Attribute-Aware Deep Hashing With Self-Consistency for Large-Scale Fine-Grained Image Retrieval," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 11, pp. 13904-13920,. 2023

[21] A. S. Das, M. Datar, A. Garg, and S. Rajaram. Google news personalization: scalable online collaborative filtering. In Proceedings of the 16th International Conference on World Wide Web. 2007, pp.271-280.

[22] H. Liu, W. Zhou, Z. Wu, S. Zhang, G. Li and X. Li, "Refining Codes for Locality Sensitive Hashing," IEEE Trans. Knowl. Data Eng., vol. 36, no. 3, pp. 1274-1284, 2024.

[23] A. Karatzoglou, M. Weimer, and A. J. Smola. Collaborative filtering on a budget. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, 2010, pp. 389-396.

[24] Zhiwei Zhang, Qifan Wang, Lingyun Ruan, and Luo Si. Preference preserving hashing for efficient recommendation. In

Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, 2014, pp.183-192.

[25] X. Liu, J. He, C. Deng, and B. Lang. Collaborative hashing. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2014, pp.2147-2154.

[26] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua, "Discrete Collaborative Filtering," in Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, Pisa Italy: ACM, Jul. 2016, pp. 325–334.

[27] Y. Zhang, D. Lian, and G. Yang, "Discrete Personalized Ranking for Fast Collaborative Filtering from Implicit Feedback," In Proceedings of the AAAI Conference on Artificial Intelligence, vol. 31, no. 1, pp. 1669–1675, 2017.

[28] C. Liu, T. Lu, X. Wang, Z. Cheng, J. Sun, and S. C. H. Hoi, "Compositional Coding for Collaborative Filtering," in Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris France: ACM, Jul. 2019, pp. 145–154.

[29] Hansen, Casper, et al. "Content-aware neural hashing for cold-start recommendation." Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval. May 2020. pp. 971-980.

[30] C. Hansen, J. G. Simonsen, and C. Lioma, "Projected Hamming Dissimilarity for Bit-Level Importance Coding in Collaborative Filtering," In Proceedings of the Web Conference 2021, in WWW '21. New York, NY, USA, 2021, pp. 261–269.

[31] H. Liu, Y. Wei, J. Yin, and L. Nie, "HS-GCN: Hamming Spatial Graph Convolutional Networks for Recommendation," IEEE Trans. Knowl. Data Eng., vol. 35, no. 6, pp. 5977–5990, 2023.

[32] X. Luo, H. Wu, Z. Wang, J. Wang and D. Meng, "A Novel Approach to Large-Scale Dynamically Weighted Directed Network Representation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 44, no. 12, pp. 9756-9773, 2022.

[33] X. Luo, Y. Zhou, Z. Liu and M. Zhou, "Fast and Accurate Non-Negative Latent Factor Analysis of High-Dimensional and Sparse Matrices in Recommender Systems," IEEE Trans. Knowl. Data Eng., vol. 35, no. 4, pp. 3897-3911, 2023.

[34] Bilal, M. Pant, H. Zaheer, L. García-Hernández, and A. Abraham, "Differential evolution: A review of more than two decades of research," Eng. Appl. Artif. Intell., vol. 90, p. 103479, 2020.

[35] J. -Y. Li, K. -J. Du, Z. -H. Zhan, H. Wang and J. Zhang, "Distributed Differential Evolution With Adaptive Resource Allocation," IEEE Trans. Cybern., vol. 53, no. 5, pp. 2791-2804, 2023.

[36] Y. Song, X. Cai, X. Zhou, B. Zhang, H. Chen, Y. Li, W. Deng, and W. Deng, "Dynamic hybrid mechanism-based differential evolution algorithm and its application," Expert Syst. Appl., vol 213, pp. 118834, 2023.

[37] I. Triguero, S. García, and F. Herrera, "SEG-SSC: A framework based on synthetic examples generation for self-labeled semi-supervised classification," IEEE Trans. Cybern., vol. 45, no. 4, pp. 622–634, 2015.

[38] D. Wu, B. Sun, and M. Shang, "Hyperparameter Learning for Deep Learning-based Recommender Systems," IEEE Trans. Services Comput., vol. 16, no. 4, pp. 2699-2712, 2023.

[39] Q. Li and M. Shang, "BALFA: A brain storm optimization-based adaptive latent factor analysis model," Inf. Sci., vol. 578, pp. 913–929, 2021.

[40] G. Xu and G. Yu, "On convergence analysis of particle swarm optimization algorithm," J. Comput. Appl. Math., vol. 333, pp. 65–73, 2018.

[41] J. Kwon, "Robust visual tracking based on variational auto-encoding markov chain monte carlo," Inf. Sci., vol. 512, pp. 1308–1323, 2020.

[42] S. Zhang, L. Yao, A. Sun, and Y. Tay, "Deep learning based recommender system: A survey and new perspectives," ACM Comput. Surv., vol. 52, no. 1, pp. 5:1–5:38, 2019.

[43] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," J. Mach. Learn. Res., vol. 7, pp. 1–30, 2006.

[44] L. Zhu, C. Zheng, W. Guan, J. Li, Y. Yang and H. T. Shen, "Multi-Modal Hashing for Efficient Multimedia Retrieval: A Survey," IEEE Trans. Knowl. Data Eng, vol. 36, no. 1, pp. 239-260, 2024.

[45] F. Li, Z. Wang, T. Wang, L. Zhu and X. Chang, "Generative Augmentation Hashing for Few-shot Cross-Modal Retrieval," IEEE Trans. Circuits Syst. Video Technol., 2025. doi: 10.1109/TCSVT.2025.3588769.

[46] W. Guan, X. Song, D. Zhou, and L. Nie, "Hashing-Based Efficient Outfit Recommendation," Advanced Multimodal Compatibility Modeling and Recommendation. Cham: Springer Nature Switzerland, 2025: 103-122.

[47] W. Guan, X. Song, D. Zhou, and L. Nie, "Hashing-Based Efficient Outfit Recommendation," Advanced Multimodal Compatibility Modeling and Recommendation. Cham: Springer Nature Switzerland, 2025: 103-122.

[48] H. Zhang, F. Luo, J. Wu, X. He, and Y. Li, "LightFR: Lightweight Federated Recommendation with Privacy-preserving Matrix Factorization," ACM Trans. Inf. Syst., vol 41, no 4, pp.90:1-28, 2023.

[49] W. Shiao, M. Ju, Z. Guo, X. Chen, E. E. Papalexakis, T. Zhao, N. Shah, and Y. Liu, "Improving Out-of-Vocabulary Hashing in Recommendation Systems," In Proceedings of the Web Conference 2025, in WWW '25. Sydney, NSW, Australia, 2025, pp. 2521–2530.

[50] Y. Chen, Y. Fang, Y. Zhang, C. Ma, Y. Hong and I. King, "Towards Effective Top-N Hamming Search via Bipartite Graph Contrastive Hashing," IEEE Trans. Knowl. Data Eng., vol. 36, no. 12, pp. 9418-9432, 2024.

[51] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian, "Rev2: Fraudulent user prediction in rating platforms," In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, WSDM, 2018, pp. 333-341.

[52] Y. Zhong, G. Li, J. Yang, H. Zheng, Y. Yu, J. Zhang, H. Luo, B. Wang, and Z. Weng, "Learning motif-based graphs for drug–drug interaction prediction via local–global self-attention," Nat. Mach. Intell., vol 6, no 9, pp.1094-1105, 2024.

[53] D. Hong, T. G. Kolda and J. A. Duersch, "Generalized Canonical Polyadic Tensor Decomposition," SIAM Review, vol. 62, no. 1, pp. 133-163, 2020.

[54] X. Xu, M. Lin, X. Luo and Z. Xu, "An Adaptively Bias-Extended Non-Negative Latent Factorization of Tensors Model for Accurately Representing the Dynamic QoS Data," IEEE Trans. Services Comput., vol. 18, no. 2, pp. 603-617, 2025.

Di Wu (Member, IEEE) received his Ph.D. degree from the Chongqing Institute of Green and Intelligent Technology (CIGIT), Chinese Academy of Sciences (CAS), China in 2019 and then joined CIGIT, CAS, China. He is currently a Professor of the College of Computer and Information Science, Southwest University, Chongqing, China. He has over 100 publications, including 38 IEEE/ACM Transactions papers on IEEE T-KDE, T-SC, T-NNLS, T-SMC, among others, and more than 10 conference papers on ICDM, AAAI, WWW, IJCAI, and ECML-PKDD, among others. He is currently serving as an Associate Editor for the journal of Neurocomputing. He is named to the Stanford's List of World's Top 2% Scientists. For more information, please visit his homepage: https://wudi1989.github.io/Homepage/.

Shihui Li received the B.S. degree in Guangxi University of Science and Technology, LiuZhou, China in 2022. She is currently pursuing her M.S. degree in Computer Sci-ence and Technology at Southwest University, Chongqing, China. Her current research interests include data mining and hash learning.

Yi He (Member, IEEE) received his Ph.D. degree in computer science from the University of Louisiana at Lafayette and a B.E. from the Harbin Institute of Technology (China) in 2020 and 2013, respectively. Dr. Yi He is an assistant professor of Computer Science at ODU. His research focus lies broadly in data mining and machine learning and specifically in online learning, data stream analytics, and graph theory. His research outcomes have appeared in top venues, e.g., AAAI, IJCAI, WWW, ICDM, SDM, TKDE, TNNLS, to name a few. He has served as multiple roles in the data mining and machine learning community, including the registration chair of ICDM 2022, session chair of ICMR 2022, and (T)PC member of AAAI'20-22, IJCAI'22, ICDM'22, CIKM'22, ECMLPKDD'22.

Xin Luo (Fellow, IEEE) received the B.S. degree in computer science from the University of Electronic Science and Technology of China, Chengdu, China, in 2005, and the Ph.D. degree in computer science from the Beihang University, Beijing, China, in 2011. He is currently a Distinguished Professor of Data Science and Computational Intelligence, and serving as the Dean of the College of Computer and Information Science, and School of Software, Southwest University, Chongqing, China. He has authored or coauthored over 400 papers (including over 190 IEEE Transactions/Journal papers) in the areas of Artificial Intelligence and Data Science, receiving 20,000+ Google Scholar citations with the H-Index of 82. Dr. Luo was the recipient of the Outstanding Associate Editor Award from IEEE Access in 2018, IEEE/CAA Journal of Automatica Sinica in 2020, and from IEEE Transactions on Neural Networks and Learning Systems in 2022-2024. He is currently serving as an Associate Editor for IEEE Transactions on Neural Networks and Learning Systems, and IEEE/CAA Journal of Automatica Sinica. His Google Scholar page is given at the link https://scholar.google.com/citations?user=hyGlDs4AAAAJ&hl=zh-CN.

Xinbo Gao (Fellow, IEEE) received the B.Eng., M.Sc., and Ph.D. degrees in signal and information pro-cessing from Xidian University, Xi'an, China, in 1994, 1997, and 1999, respec-tively. From 1997 to 1998, he was a Research Fellow with the Department of Computer Science, Shizuoka University, Shizuoka, Japan. From 2000 to 2001, he was a Post-Doctoral Research Fellow with the Department of Information Engi-neering, The Chinese University of Hong Kong, Hong Kong. Since 2001, he has been with the School of Electronic Engineering, Xidian University. He is currently a Cheung Kong Professor of Ministry of Education, a Professor of Pattern Recognition and Intelligent System, and the Director of the State Key Laboratory of Integrated Services Networks, Xi'an. His current research interests include multimedia analysis, computer vision, pattern recognition, machine learning, and wireless communications. He has published five books and around 200 technical articles in refereed journals and proceedings. He is on the Editorial Boards of several journals, including Signal Processing (Elsevier) and Neurocomputing (Elsevier). He has served as the General Chair/Co-Chair, the Program Committee Chair/Co-Chair, or a PC Member for around 30 major international conferences. He is also a fellow of the Institution of Engineering and Technology.